DEPARTAMENTO DE INFORMÁTICA

PROYECTO FIN DE CARRERA

Ingeniería Técnica en Informática de Gestión

Análisis, Diseño e Implementación de una Aplicación Móvil para la gestión de dispositivos hardware y software en pymes

Autor: José García López

Tutor: Fuensanta Medina Domínguez

Título: Análisis, Diseño e Implementación de una Aplicación Móvil para la gestión de dispositivos hardware y software en pymes

Autor: José García López

Director: Fuensanta Medina Domínguez

EL TRIBUNAL

Presidente: Luis García Sánchez

Vocal: Maribel Sánchez Segura

Secretario: Germán Lenin Dugarte

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 24 de octubre de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de 9 - SOBRESALIENTE.

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mi padre, Jose, y a mi hermano, Vicente, porque aunque tengamos siempre nuestros más y nuestros menos, siempre han estado ahí.

A mis abuelos, Carlota y Vicente y Josefa y José, que ninguno pudo llegar a ver este momento, pero a quienes siempre tengo presentes.

A todos aquellos compañeros de clase que han luchado a mi lado por conseguir nuestra meta común de sacarnos la carrera, en especial a Joaquín Giráldez, Iván Consuegra, Alberto Uceda, Esther García, Alfredo Barrio, Luis Castro y Lourdes Crespo. Por todas esas horas de esfuerzo, de trabajo en equipo, de biblioteca, de laboratorio, de pasillos y de cafetería.

A aquellos profesores que han sido más profesores, transmitiendo su entusiasmo: Iván López Montalbán, Araceli Sanchis de Miguel, Miguel Gutiérrez, Benjamín Ramos Álvarez, Alejandro Calderón, Carlos Rascón, Ainhoa Erkoreka y en especial a Fuensanta Medina Domínguez y Maribel Sánchez Segura por responder a mi llamada de auxilio y ayudarme y ofrecerme su colaboración y con ello darme la oportunidad de terminar algo que, afortunadamente, al final he podido conseguir.

Y, por supuesto, a Conchy Parra Wic, la persona más importante de mi vida, que no sólo está conmigo en los buenos momentos, sino que también me aguanta en los malos. Y eso, con mi genio, es mucho decir. Ella también ha sufrido el esfuerzo de realizar este proyecto. Gracias por apoyarme y estar ahí siempre, a mi lado.

Resumen

El proyecto realizado muestra el desarrollo de una aplicación para dispositivos móviles. Se comienza por el estudio del mercado actual, otras herramientas ya existentes y la definición de la plataforma a utilizar a partir de los datos obtenidos. A continuación se lleva a cabo la gestión del proyecto, paso previo a la adquisición de los requisitos necesarios para realizar el análisis, el diseño y la posterior implementación de la aplicación, detallando aquellos puntos más relevantes de su realización. En esta documentación se detallan todos los pasos seguidos para completar el proyecto.

Abstract

This project shows the development of an application for mobile devices. It begins with a research of the current market, other existing tools and the definition of the platform to be used based on the obtained data. Then the project management is carried out, prior to the acquisition of the necessary requirements for the analysis, design and subsequent implementation of the application, detailing the most relevant points of its realization. All the steps taken to complete the project are detailed in this document.

Índice general

Índice de ilustraciones

Índice de tablas

Capítulo 1. Introducción y objetivos

En este capítulo se describe en qué consiste este proyecto, los motivos por los que se crea y sus objetivos. Se comenzará con la definición del problema actual y se determinarán los métodos para solventarlo.

1.1. Contextualización

Según el Directorio Central de Empresas (DIRCE), a 1 de enero del año 2015 hay en España 3.182.321 empresas, de las cuales 3.178.408 (99,9%) son PYME (entre 0 y 249 asalariados).

	Micro empresas sin asalariados	Micro empresas (1 – 9 empleados)	Pequeñas empresas (10 – 49 empleados)	Medianas empresas (50 – 249 empleados)	PYME (0 – 249 empleados)	Grandes empresas (>=250 empleados)	Total
Número de empresas	1.751.964	1.297.861	110.086	18.497	3.178.408	3.913	3.182.321
% respecto al total	55,1	40,8	3,5	0,6	99,9	0,1	100

TABLA 1. EMPRESAS SEGÚN ESTRATO DE ASALARIADOS Y PORCENTAJE TOTAL EN ESPAÑA

Esto indica que en España la gran mayoría de las empresas tiene menos de 10 empleados. Coincide además que las micro empresas son las que cuentan con menos recursos tecnológicos y, por tanto, no soportan su negocio tanto en el uso de ordenadores, conexión a internet y dispositivos conectados.

Todo indica que existe posibilidad de penetración en un mercado en el que todavía hay mucho por hacer como es el de la Tecnología de la Información para Pymes, puesto que

éstas deben adaptarse en un mundo cada vez más competitivo, reduciendo costes y aumentando su rendimiento que, en definitiva, es lo que aumenta la rentabilidad y la competitividad.

Con la vista puesta en estas pequeñas empresas surge un proyecto sencillo, pero a la vez potente y personalizable según sus necesidades.

La idea inicial surgió de la experiencia de trabajo en una pequeña empresa dedicada al mantenimiento informático de empresas. Dicha empresa realiza todo tipo de soporte, ya sea telefónicamente, utilizando conexiones remotas con los clientes para solucionar incidencias o desplazándose a sus instalaciones para solventar cualquier problema hardware.

Cuando se da de alta un nuevo cliente, éste ha contratado un mantenimiento que básicamente se ha presupuestado midiendo la cantidad y el tipo de los dispositivos a los que se puede dar soporte o mantenimiento. El perfil de cliente es muy variado. Desde micro empresas sin asalariados que disponen únicamente de un portátil, una impresora y un *smartphone*, a empresas que disponen de equipos servidores, con varios puestos cliente y sus respectivas *tablets*, *smartphones*, impresoras de red o impresoras monopuesto.

Esto, cuando la cartera de clientes tiene un tamaño manejable para la cantidad de personal disponible, no supone un problema. Posiblemente, basta un pequeño ejercicio de memoria para, por ejemplo, recordar qué impresora y cómo la tiene conectada y configurada un determinado cliente. En el peor de los casos, se consulta el programa de facturación para revisar las últimas impresoras que se le vendieron al cliente. Pero el problema se acrecienta a medida que la empresa va adquiriendo nuevos contratos de mantenimiento, nuevos clientes, los cuales a su vez tienen cada vez más dispositivos, ya sean anteriores al contrato o posteriores, los cuales en ocasiones pueden sufrir incidencias, reparaciones, sustituciones o renovaciones. Los técnicos encargados de resolver las incidencias deben desplazarse al cliente y resolver el problema, pero surgen ciertos problemas derivados de esta falta de control: en ocasiones el técnico asignado

no siempre es el mismo para un determinado cliente, con lo que un determinado técnico desconoce qué se realizó en la última intervención por parte de otro técnico; en ocasiones, el técnico es el mismo, pero no recuerda qué acciones se tomaron anteriormente ante un determinado problema. Todo parece indicar que el llevar un registro exhaustivo de dispositivos de los clientes, sus incidencias y sus soluciones podría solventar muchas situaciones en las que el técnico se ve obligado a parar su trabajo para realizar llamadas, consultar a la central o a otros compañeros o a emplear tiempo excesivo en hacer memoria de cómo resolvió ese mismo problema en ese mismo cliente varios meses atrás.

La idea es sencilla: un registro exhaustivo de los dispositivos y de las intervenciones realizadas en ellos, así como datos clave que puedan facilitar el trabajo diario de los técnicos, tales como la fecha de la extinción de la garantía de dichos dispositivos. Sólo requiere el tiempo que se tardaría en rellenar un parte de trabajo, pero esa información estaría siempre a mano para cualquiera que consultara los datos. Obviamente, aquí entra en juego la informática. Una pequeña base de datos con información de dispositivos, sus números de serie, su garantía, trabajos realizados de reparación si los hubiese y que además fuese fácilmente consultable por los técnicos. Etiquetando cualquier dispositivo con un código de barras y leyéndolo para obtener toda su información disponible parece el camino a seguir. Y qué mejor manera de hacerlo que con una herramienta que, sin saberlo, hace tiempo que tienen en sus manos y llevan a todas partes consigo: los teléfonos móviles. Su cámara, aparte de hacer fotografías, se puede utilizar para leer códigos de barra tan rápida y cómodamente como lo hace cualquier lector láser de supermercado. Esto, combinado con una pequeña base de datos, puede facilitar mucho el trabajo de los técnicos, ahorrando tiempo y por lo tanto, dinero. El coste para la empresa es únicamente la de adquirir el software. La herramienta hardware ya la tenían. Así nace la aplicación DeviRec.

1.2. Objetivos y Alcance

El objetivo de este proyecto es la creación de una app para smartphones que lleve un registro de dispositivos al cual se acceda introduciendo un código de barras ya sea leído a través de la cámara del smartphone o introduciéndolo manualmente en aquellos casos en los que la lectura del código de barras resulte complicada por diversos motivos (luminosidad, deterioro de la etiqueta, etc.). Del mismo modo se podrían dar de alta nuevos dispositivos leyendo códigos que no se encuentren previamente en la base de datos, modificar los ya existentes añadiendo nuevos datos o eliminar alguno cuyo dispositivo al que pertenece haya sido desechado por obsolescencia o avería irreparable.

Para alcanzar este objetivo y tener posibilidades de éxito, se debe realizar teniendo en cuenta el entorno y el perfil del cliente. Es por ello que previamente se va realizar un estudio de las diferentes plataformas móviles disponibles en el mercado y se tomará una decisión sobre en cuál de ellas implementar la app en primera instancia, de modo que se maximicen los resultados apostando por invertir los recursos en una única plataforma, la cual debe ser la más apta. La elección de la plataforma no viene determinado únicamente por la más adecuada dentro de las diferentes plataformas disponibles, sino que hay que tener en cuenta a quién va dirigido el producto, el perfil del cliente que utilizará la aplicación. Para ello se definirá el perfil de cliente que usará el producto así como sus preferencias de plataformas móviles existentes y con ello se obtendrá un mejor afinamiento en la obtención de resultados.

1.3. Fases del desarrollo del proyecto

Las fases para realizar el proyecto son las siguientes:

- Análisis del entorno: analizar el entorno o contexto donde se va a trabajar.
    - Análisis del mercado: definir cuál es el perfil del cliente que utilizará la aplicación y el tipo de aplicación móvil adecuada.
    - Análisis de las plataformas móviles disponibles en el mercado: estudio de las diferentes plataformas en las cuales se podría implementar una aplicación para el cliente para determinar cuál es la más adecuada para el proyecto.
- Gestión de proyecto: planificar, organizar, motivar y controlar los recursos para la realización del proyecto, determinando también las limitaciones de alcance, tiempo, calidad y presupuesto.
- Proyecto:
    - Análisis y especificación de requisitos: llevar a cabo un estudio de las necesidades del cliente que se quieren cubrir, a partir de los cuales se realiza la documentación de los requisitos.
    - Diseño: descomponer y organizar el sistema en elementos que puedan elaborarse por separado, dando como resultado el SSD (Documento de Diseño de Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como el modo en que se combinan unas con otras.
    - Implementación: a partir del diseño se realiza la implementación de la aplicación en la plataforma seleccionada.
    - Pruebas: batería de pruebas para con las que se depura la aplicación, para detectar posibles errores y obtener una versión lo más estable posible.
    - Implantación y mantenimiento: instalar la aplicación en el cliente, configuración y formación, así como mantenimiento y soporte técnico.

- Memoria PFC: documento que contiene toda la información del Proyecto Fin de Carrera.
- Presentación PFC: defensa del PFC ante el tribunal de evaluación.

En la Ilustración 1 se muestra un resumen más gráfico e intuitivo de todo el proceso.

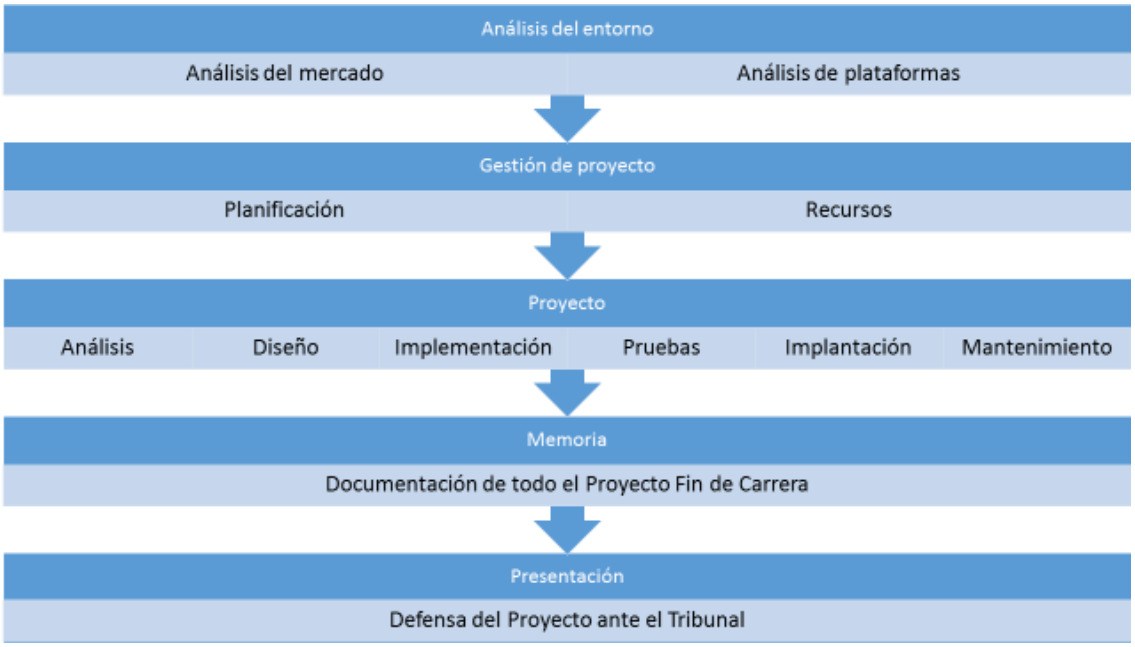


ILUSTRACIÓN 1. FASES DEL DESARROLLO DEL PROYECTO

1.4. Herramientas utilizadas

Para llevar a cabo el proyecto se han utilizado las siguientes herramientas:

- Para la elaboración de la documentación:
  - Microsoft Word 2013 para la documentación global.
  - Microsoft PowerPoint 2013 como herramienta auxiliar para la realización de gráficos, ilustraciones y esquemas visuales.

- Para la elaboración de la aplicación:

  - Android Studio 2.1 como herramienta de programación principal.
  - Java SE Development Kit 7 como software proveedor de herramientas de desarrollo para la programación en Java. Requisito de Android Studio para su correcto funcionamiento.
- Equipos hardware:
  - Ordenador portátil con Windows 10.
  - Ordenador portátil Mac con OSX El Capitán 10.11.3
  - Dispositivos de pruebas:
    - Smartphone BQ M5 con Android Marshmallow 6.0.1
    - Smartphone LG G4 con Android Marshmallow 6.0
    - Smartphone BQ Aquaris 5 con Android Kit Kat 4.4.2
    - Smartphone Samsung Galaxy Note 2 con Android Kit Kat 4.4.2
    - Smartphone Samsung Galaxy Young con Android Jelly Bean 4.2
    - Tablet BQ Edison 3 con Android Kit Kat 4.4.2
- Otras herramientas:
  - Google Drive como herramienta de copia de seguridad.
- Documentación (ver Bibliografía).

1.5. Estructura de la memoria

La estructura del presente documento se expone a continuación, explicando brevemente el contenido de cada capítulo:

- Capítulo 1: Introducción y objetivos

  Se realiza una introducción poniendo en contexto cuál es el problema, qué se pretende realizar para resolverlo y cómo se ha llegado a su resolución.

- Capítulo 2: Estado de la cuestión

  Se detalla cuál es la situación actual antes del comienzo del proyecto. Se estudian las plataformas disponibles y se estudia el perfil del cliente que utilizará la aplicación y con ello se busca la forma más adecuada de afrontar el problema.

- Capítulo 3: Gestión de proyecto

  Se lleva a cabo una planificación del proyecto, especificando plazos, recursos, organización y estimación de costes en base a todo ello.

- Capítulo 4: Desarrollo de la herramienta

  Se lleva a cabo la implementación del proyecto mediante la aplicación del ciclo de vida de software desde el análisis de requisitos a la puesta en marcha en el cliente.

- Capítulo 5: Conclusiones

  Tras todo el proceso detallado en los capítulos previos, se extraen todas las conclusiones obtenidas a través de la experiencia de la realización del proyecto y se comparan con las ideas iniciales.

- Capítulo 6: Líneas futuras

  Como todo proyecto, siempre existe la posibilidad de ser ampliado y/o mejorado. En este apartado se establecen líneas a seguir a partir del alcance actual de este proyecto, para obtener versiones más completas y complejas.

- Capítulo 7: Bibliografía y referencias

  Se especifican las publicaciones referenciadas a través de este documento, así como cualquier otra fuente que se haya consultado que haya servido de apoyo para la finalización de este proyecto.

Capítulo 2. Estado de la cuestión

2.1. Introducción

El objetivo de este proyecto es la creación de una aplicación para dispositivos móviles que permita al usuario tener control e información sobre una gran cantidad de dispositivos desplegados en sus diferentes clientes. Para ello se utilizará una identificación única mediante un código de barras asignado a todos y cada uno de los dispositivos de modo que, leyendo su código mediante la cámara de su dispositivo móvil, se pueda rápidamente acceder a todos los datos necesarios.

Esta aplicación, por tanto, deberá poder leer códigos de barra aprovechando la cámara de un teléfono móvil o Tablet, devolviendo como resultado toda la información disponible sobre el dispositivo previamente guardada o permitir el alta de un nuevo dispositivo si el código leído no correspondiese a ningún dispositivo de la base de datos. Otras operaciones disponibles deben ser la modificación de la información o el borrado en caso de la baja del dispositivo al que pertenece el código. Así mismo, ampliando su usabilidad, se permite la introducción manual de códigos en previsión de aquellos casos en los que el código resulte ilegible por la cámara.

Como paso previo, se realizará un estudio de mercado, examinando las aplicaciones ya existentes en el mercado. Posteriormente, se pasará a estudiar la plataforma a utilizar para su implementación y la realización de un perfil de cliente que utilizará la aplicación. Ambos resultados, interconectados entre sí, determinarán a la vez la plataforma a utilizar y el cliente que utilizará la aplicación. Esto se debe a que la elección de una plataforma no se va a restringir a sopesar sus ventajas e inconvenientes respecto a otras, sino que se tendrá en cuenta también a qué cliente va dirigido el producto, con el objetivo de encontrar la solución más rentable.

2.2. Aplicaciones ya existentes en el mercado

Antes de comenzar un proyecto para la realización de una aplicación, es muy recomendable observar lo ya existente en el mercado. Es muy posible que ya existan aplicaciones que tengan la misma funcionalidad, lo cual puede determinar si merece la pena emplear dinero y recursos en la creación de una aplicación que hace lo mismo. Se van a examinar las aplicaciones de la competencia y se van a comparar con el proyecto que se pretende realizar, viendo sus ventajas y deficiencias.

2.2.1. GLPi

La aplicación GLPi [1] permite construir una base de datos con inventario de los recursos de la empresa (ordenadores, software, impresoras…). Tiene funciones mejoradas para hacer la vida del administrador más fácil, con sistema de trazabilidad, notificaciones por correo electrónico y métodos para construir una base de datos con la información básica sobre la topología de red.

Las funcionalidades principales son:

1. Inventario preciso de todos los recursos técnicos. Todas sus características se almacenarán en la base de datos.
2. Gestión e historial de las actuaciones de mantenimiento y los procedimientos vinculados. Es una aplicación dinámica y está directamente conectada a los usuarios, quienes pueden hacer peticiones a los técnicos. Una interfaz autoriza al usuario a solicitar servicio sobre alguno de los recursos para los que tenga acceso.

Se trata de una aplicación muy completa que se debe instalar en un servidor con requisitos tales como:

- Apache 2 o superior / Microsoft IIS
- PHP versión 5.3 o superior.
- Extensiones PHP:
    - JSON
    - Mbstring
    - MySQL
    - Session
- Extensiones PHP opcionales pero recomendables:
    - CLI.
    - CURL.
    - DOMXML.
    - GD.
    - IMAP.
    - LDAP.
    - OpenSSL.

Como se puede apreciar en la Ilustración 2, se puede especificar con todo detalle cada característica de cualquier dispositivo o recurso. En el ejemplo, un servidor en el que se especifican desde la placa base a la fuente de alimentación, pasando por la memoria RAM o el disco duro que lleva, técnico asignado, número de serie, etc.

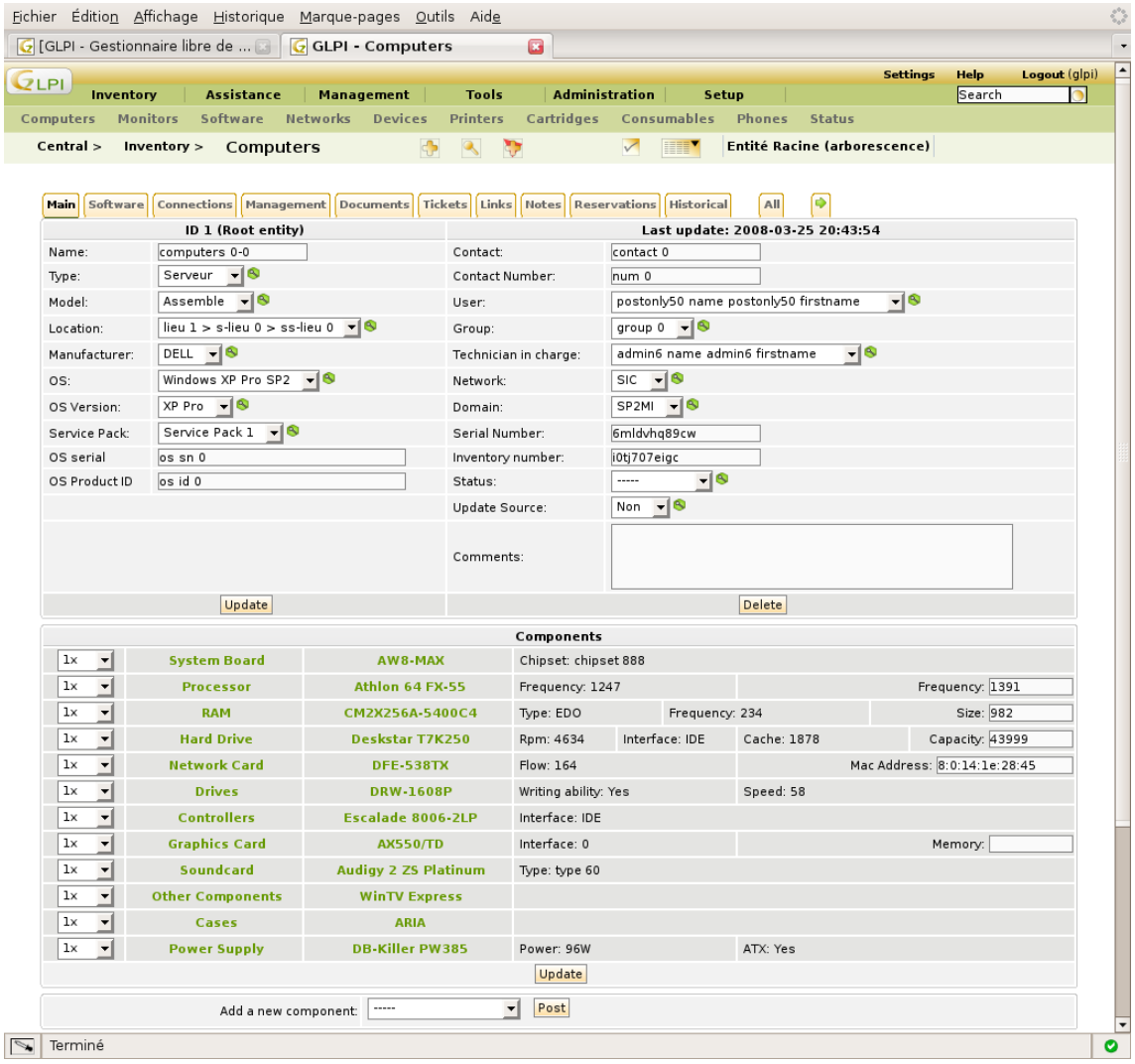
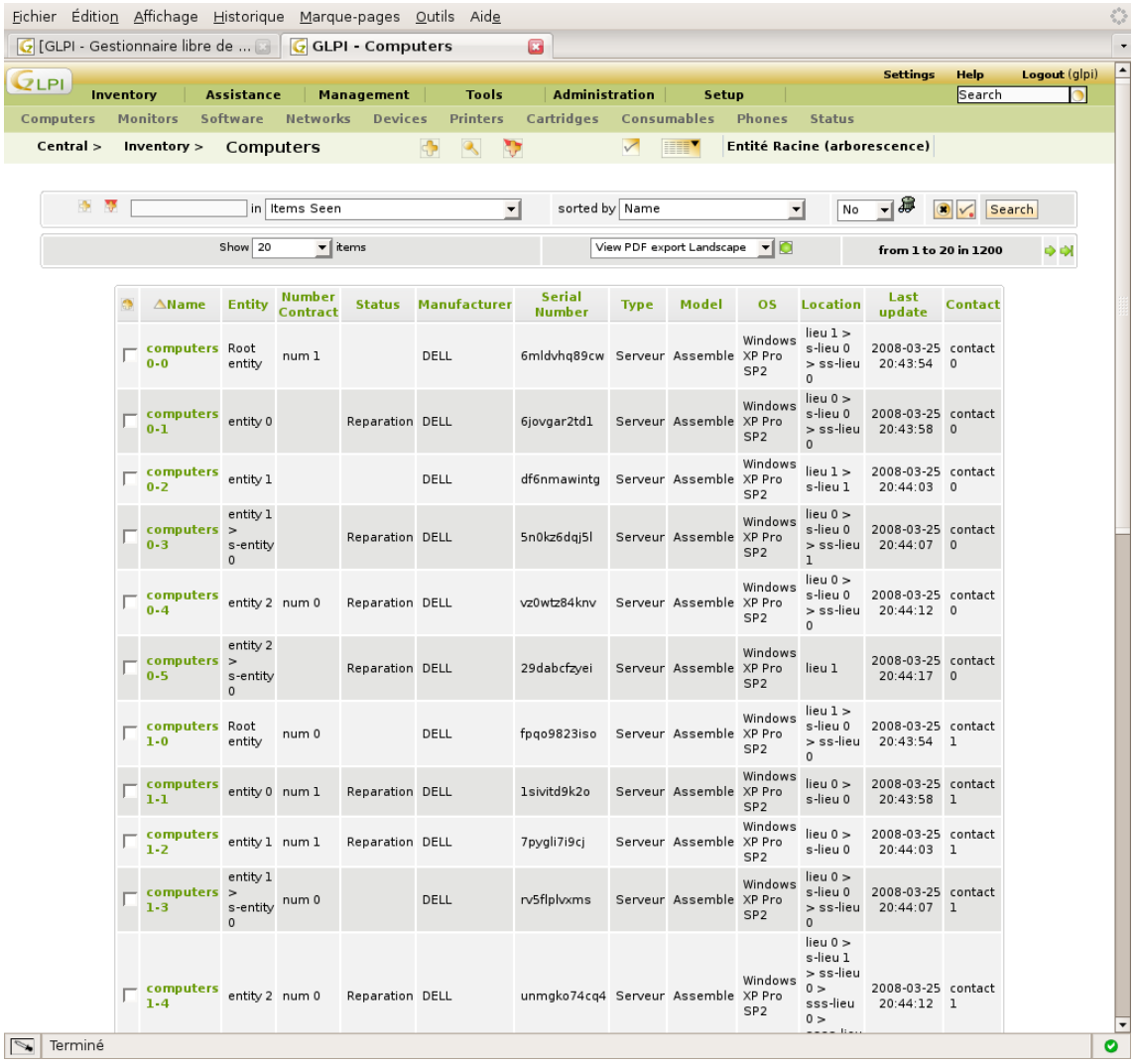


ILUSTRACIÓN 2. EJEMPLO DE TODOS LOS COMPONENTES DE UNA COMPUTADORA EN GLPI

Una vez detallados todos los dispositivos, como muestra la Ilustración 3 se puede obtener un listado completo en una vista más general, a modo de inventario, únicamente con los detalles más importantes como el número de serie, el estado en el que se encuentra cada dispositivo, su sistema operativo, última actualización, fabricante, etc.



Name	Entity	Number Contract	Status	Manufacturer	Serial Number	Type	Model	OS	Location	Last update	Contact
computers 0-0	Root entity	num 1		DELL	6mldvhq89cw	Serveur	Assemble	Windows XP Pro SP2	lieu 1 > s-lieu 0 > ss-lieu 0	2008-03-25 20:43:54	contact 0
computers 0-1	entity 0		Reparation	DELL	6jovgar2td1	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0 > ss-lieu 0	2008-03-25 20:43:58	contact 0
computers 0-2	entity 1			DELL	df6nmawintg	Serveur	Assemble	Windows XP Pro SP2	lieu 1 > s-lieu 1	2008-03-25 20:44:03	contact 0
computers 0-3	entity 1 > s-entity 0		Reparation	DELL	5n0kz6dqj5l	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0 > ss-lieu 1	2008-03-25 20:44:07	contact 0
computers 0-4	entity 2	num 0	Reparation	DELL	vz0wtz84knv	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0 > ss-lieu 0	2008-03-25 20:44:12	contact 0
computers 0-5	entity 2 > s-entity 0		Reparation	DELL	29dabcfzyei	Serveur	Assemble	Windows XP Pro SP2	lieu 1	2008-03-25 20:44:17	contact 0
computers 1-0	Root entity	num 0		DELL	fpqo9823iso	Serveur	Assemble	Windows XP Pro SP2	lieu 1 > s-lieu 0 > ss-lieu 0	2008-03-25 20:43:54	contact 1
computers 1-1	entity 0	num 1	Reparation	DELL	1sivitd9k2o	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0	2008-03-25 20:43:58	contact 1
computers 1-2	entity 1	num 1	Reparation	DELL	7pygli7i9cj	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0	2008-03-25 20:44:03	contact 1
computers 1-3	entity 1 > s-entity 0	num 0		DELL	rv5flplvxms	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 0 > ss-lieu 0	2008-03-25 20:44:07	contact 1
computers 1-4	entity 2	num 0	Reparation	DELL	unmgko74cq4	Serveur	Assemble	Windows XP Pro SP2	lieu 0 > s-lieu 1 > ss-lieu 0 > sss-lieu 0 >	2008-03-25 20:44:12	contact 1

ILUSTRACIÓN 3. INVENTARIO EN GLPI

Adicionalmente cuenta con una interfaz para la comunicación de las incidencias por parte de los usuarios, mostrada en la Ilustración 4. Éstos pueden reportar cualquier problema con los dispositivos que utilizan y el sistema permite usar un calendario para reservar citas, así como realizar una labor de trazabilidad de los trabajos realizados y pendientes.

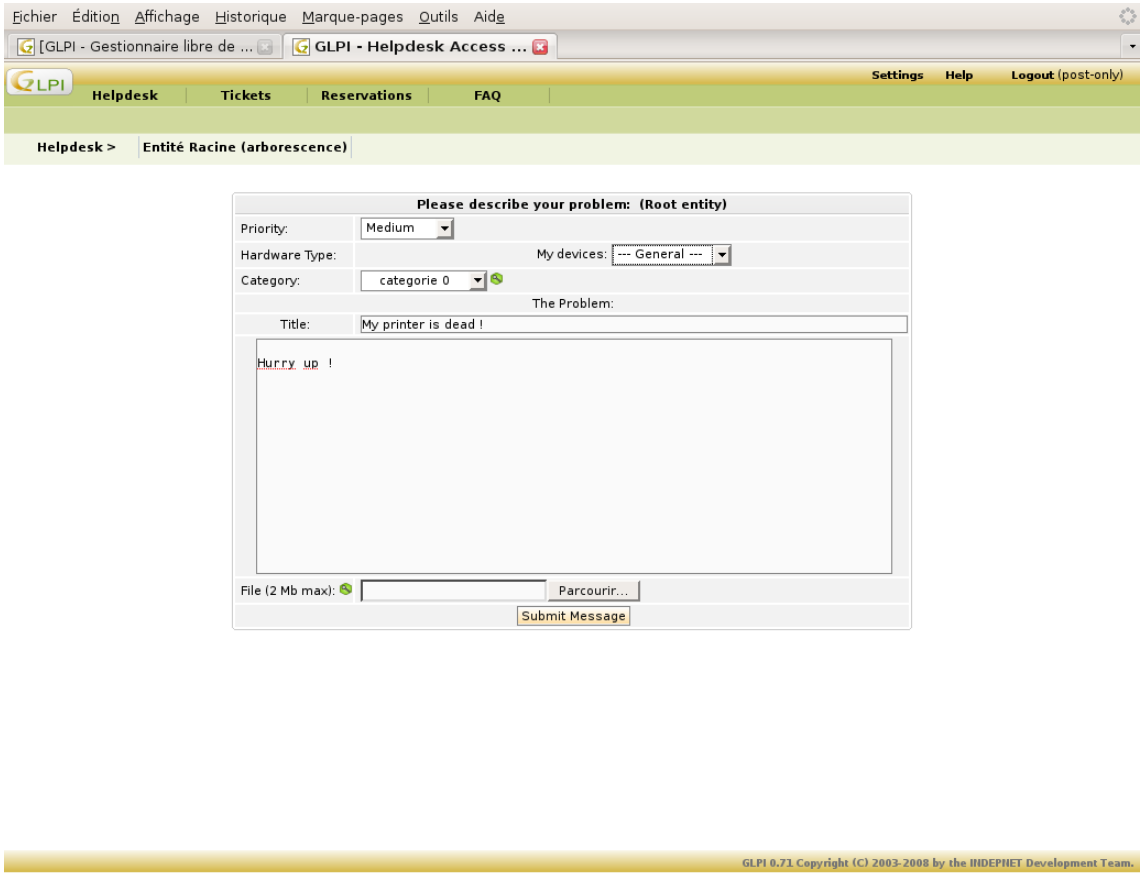


ILUSTRACIÓN 4. INTERFAZ PARA LA COMUNICACIÓN DE INCIDENCIAS DE GLPI

2.2.1.1. Ventajas de GLPi

GLPi posee las siguientes características positivas:

- Permite realizar un inventario completo y detallado de una empresa.
- Permite realizar la topología de red.
- Interfaz para comunicación de incidencias de los usuarios de la empresa.
- Interfaz de gestión de las incidencias (agenda, seguimiento de tareas, etc.)

2.2.1.2. Inconvenientes de GLPi

En el apartado de inconvenientes de esta aplicación, se encuentran los siguientes:

- Se trata de un software para servidor.
- La interfaz está pensada para su utilización mediante navegador y a través de un ordenador (sobremesa o portátil), así como de una conexión al servidor.
- Dispone de una app para Android, pero únicamente de la gestión de incidencias (ver los propios tickets, ver otros tickets, crear tickets).

2.2.2. Retail Inventory

Retail Inventory [2] es una aplicación para iOS que permite leer códigos de barra para llevar a cabo el inventario de un negocio. Se puede utilizar la cámara de un iPad, iPod o iPhone para ir leyendo los códigos. Una vez almacenados, los datos se pueden enviar por correo electrónico o vincular a otra aplicación, Cashier Live, también para iPad, iPod o iPhone, que es la parte TPV (Terminal Punto de Venta) para el negocio. La Ilustración 5 muestra su interfaz.

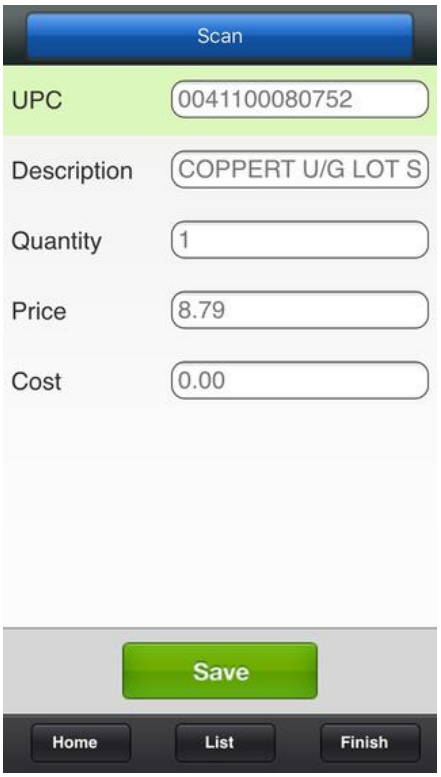


ILUSTRACIÓN 5. INTERFAZ DE RETAIL INVENTORY

2.2.2.1. Ventajas de Retail Inventory

Las ventajas de Retail Inventory en su utilización son:

- Aplicación muy sencilla que lee directamente códigos de barras.
- Permite realizar inventarios rápidamente.
- Con su complemento Cashier Live permite la gestión de pequeños negocios cómodamente.
- Muy manejable al poderse utilizar en dispositivos móviles.

2.2.2.2. Inconvenientes de Retail Inventory

Los inconvenientes de Retail Inventory son los siguientes:

- Aplicación exclusiva para iOS.

- La lectura de códigos sirve únicamente para realizar inventario.
- Sin Cashier Live, la aplicación no es más que una contadora de artículos.

2.2.3. ABC Inventory

ABC Inventory [3] es un software para Windows que permite un control de inventario exhaustivo, rozando el nivel de un software ERP (*Enterprise Resource Planning,* Planificación de Recursos de Empresa en español), permitiendo una gestión de stock en la que se pueden configurar notificaciones si el stock está por debajo de un determinado mínimo, realizar pedidos de compra y sacar informes.

En la Ilustración 6, se observa que además de un inventario normal, se pueden realizar escandallos, es decir, determinar si uno o más artículos forman parte de otro artículo. En caso de realizar una entrada o salida del artículo, se realizarían igualmente las entradas o salidas correspondientes en cantidad de cada artículo dependiente.

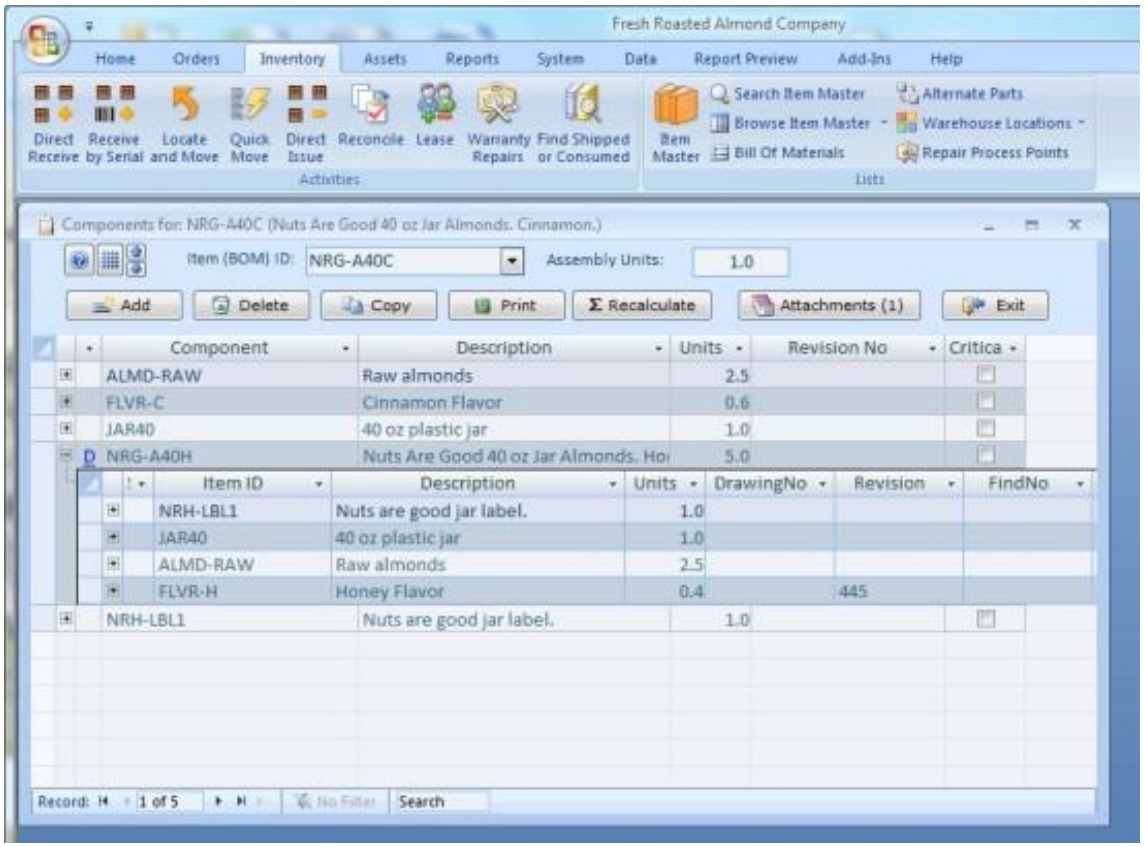


ILUSTRACIÓN 6. ESCANDALLO EN ABC INVENTORY

Forma parte de una suite más grande llamada Almyta Control System, que es un software MRP (*Material Requirements Planning*, Planificación de Requisitos de Materiales en español), aunque se puede utilizar como módulo completamente independiente o bien integrado en Almyta. Así mismo, dispone de opciones para mostrar listados e informes, tal y como se muestra en la Ilustración 7.

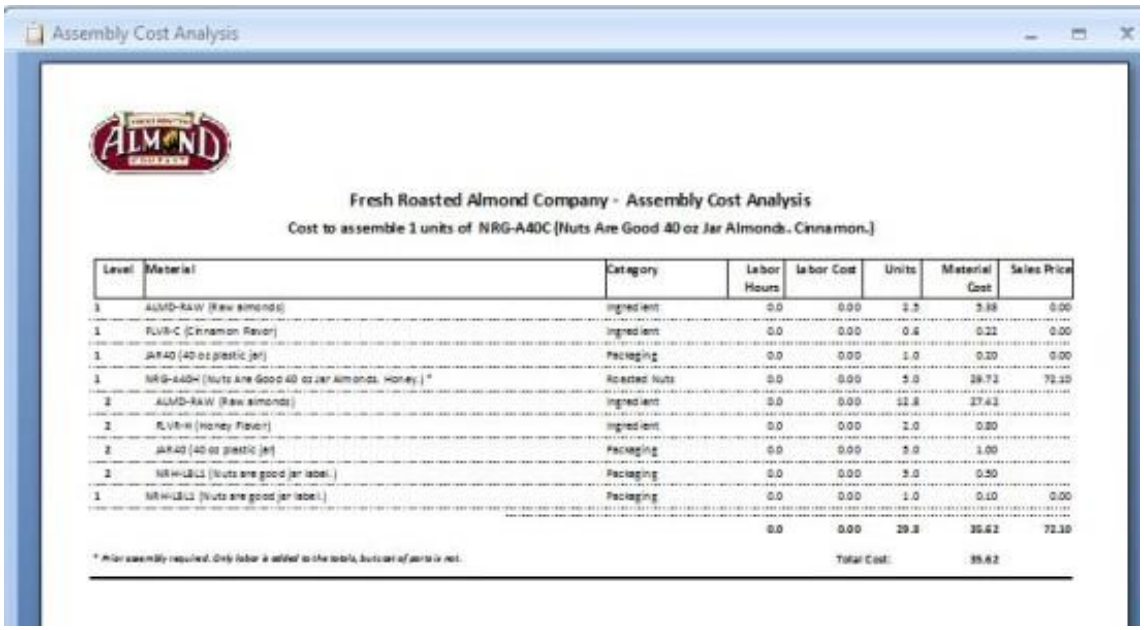


ILUSTRACIÓN 7. GENERACIÓN DE INFORMES EN ABC INVENTORY

2.2.3.1. Ventajas de ABC Inventory

Las ventajas de ABC Inventory respecto a su competencia son:

- Sencillo sistema de inventario para Windows.
- Posibilidad de realizar escandallos.
- Permite sacar listados.
- Posibilidad de ampliar la funcionalidad adquiriendo Almyta.

2.2.3.2. Inconvenientes de ABC Inventory

ABC Inventory posee las siguientes desventajas:

- Disponible únicamente para Windows.
- No permite añadir información adicional al artículo introducido.
- No tiene gestión de incidencias.

2.2.4. PartKeepr

PartKeepr [4] es una sencilla aplicación de gestión de inventario para componentes electrónicos. Al ser tan específica no es tan completa como otras ya mencionadas. Se requiere tener configurada previamente una base de datos en MySQL. En este caso se puede añadir información adicional a los artículos del inventario, además de distribuidores, fabricantes y los adjuntos que se necesiten. Adicionalmente se pueden añadir tantos parámetros de un artículo como se necesiten y posteriormente realizar búsquedas por dichos parámetros, ordenar por esas nuevas columnas, etc. En la Ilustración 8 se puede ver cómo es la pantalla de introducción de los detalles asociados a un artículo.

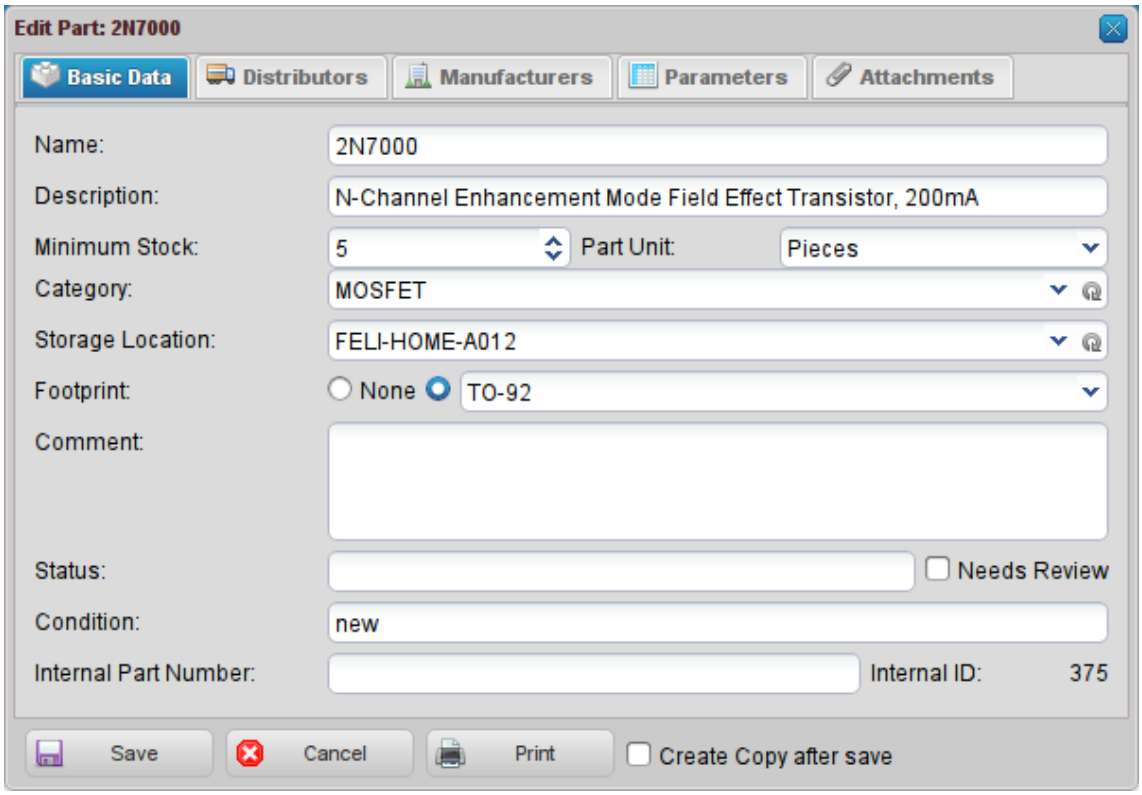


ILUSTRACIÓN 8. FICHA DE UN ARTÍCULO EN PARTKEEPR

2.2.4.1. Ventajas de PartKeepr

PartKeepr posee las siguientes características que la hacen valer sobre la competencia:

- Es software de código abierto, bajo licencia GPLv3.
- Es sencillo pero muy práctico para el sector al que está enfocado.
- Fácilmente modificable para agregar nuevos parámetros y utilizarlos en las búsquedas.

2.2.4.2. Inconvenientes de PartKeepr

Los inconvenientes de PartKeepr son los siguientes:

- Se encuentra disponible sólo para sistemas tipo Unix (Linux, FreeBSD).
- No tiene gestión de incidencias sobre los artículos.

2.3. Resumen comparativo de las herramientas del mercado

Vistas diferentes herramientas que ya existen en el mercado con sus pros y sus contras, a continuación se muestra una tabla resumen en la que se evalúan ciertas características, planteadas como objetivo de la aplicación a implementar en este proyecto, especificando si las herramientas actuales las cumplen o no.

A modo de recordatorio, de forma muy resumida, se requiere una aplicación para smartphones, la cual se llamará DeviRec, que lleve un control de los dispositivos disponibles, leídos mediante códigos de barras, permitiendo guardar detalles técnicos de interés sobre ellos.

	GLPi	Retail Inv.	ABC Inv.	PartKeepr	DeviRec
Nativa Smartphone	✗	✓	✗	✗	✓
Inventario completo detallado	✓	✗	✓	✓	✓
Lectura códigos portátil	✗	✓	✗	✗	✓

TABLA 2. RESUMEN DE APLICACIONES DEL MERCADO

Como se puede apreciar en la Tabla 2, cada una de las aplicaciones ha ido llenando un hueco de los requisitos. Sólo una en iOS para dispositivos móviles como tablets o smartphones que permita movilidad, pero con el gran inconveniente de que sirve únicamente para contar unidades y llevar un inventario, sin más, necesitando de una aplicación adicional enfocada al comercio. Por ello, existe un vacío que se puede ocupar en dicha tabla y que sí que cumpliría DeviRec.

2.4. App: La aplicación para dispositivos móviles

Una app es una aplicación de software que se instala en dispositivos móviles o tablets para ayudar al usuario en una labor concreta, ya sea de carácter profesional o de ocio. Es decir, se suele referenciar como “aplicación” a los programas clásicos, mientras que su versión para dispositivos móviles se suele llamar “app”. Más allá de un simple cambio de nombre, existe algo más. En primer lugar, una app debe ser sencilla de utilizar con los dedos en pantallas táctiles, contará con un teclado en pantalla sólo en el momento en el que haga falta escribir algo y la interfaz debe tener en cuenta que no habrá, por ejemplo, un ratón para acceder a los menús o seleccionar opciones. En segundo lugar, pero no menos importante, una app puede ser cerrada en cualquier momento, por el usuario o por el sistema, lo cual implica ciertos cambios a la hora de su programación que la diferencian de una aplicación de escritorio clásica.

2.5. Tipos de apps

A nivel de programación existen varias formas de desarrollar una app. Cada una de ellas tiene sus ventajas y sus limitaciones y en gran medida van a condicionar el diseño visual y la interacción con el usuario. Actualmente hay tres tipos de aplicaciones: apps o aplicaciones nativas, web apps o aplicaciones web y aplicaciones híbridas.

2.5.1. Aplicaciones nativas

Las aplicaciones nativas han sido desarrolladas con el software que ofrece cada sistema operativo para los programadores, un conjunto de herramientas de desarrollo de software llamado genéricamente *Software Development Kit* o SDK.

De este modo, las plataformas Android, iOS y Windows Phone tienen un SDK diferente y las aplicaciones nativas se diseñan y programan específicamente para cada plataforma, en el lenguaje utilizado por el SDK.

Este tipo de apps se suele descargar e instalar desde las tiendas de apps, con lo cual se puede sacar partido a las diferentes herramientas de promoción y marketing de las que se dispone en cada una de ellas.

Si una app se actualiza, el usuario tiene que descargarla de nuevo para obtener la última versión. No se pueden aplicar los clásicos parches a una app. Cuando se localiza un fallo de software o *bug,* se soluciona, se recompila la app y se pone a disposición de los usuarios para que sustituyan la versión anterior.

Las principales ventajas de las aplicaciones nativas son:

- Pueden utilizar las notificaciones del sistema operativo para avisos importantes al usuario, incluso sin estar utilizando la app.

- Están integradas en el dispositivo, por lo que no requieren internet para funcionar, y se les permite utilizar todo el hardware tal como la cámara, los sensores GPS, acelerómetro, giroscopio, etc.
- A nivel de diseño, la interfaz está basada en las guías de cada sistema operativo, con lo que se logra una mayor coherencia con el resto de aplicaciones y con el propio sistema operativo. Esto favorece al usuario porque se encuentra con interfaces familiares que cambian poco de una aplicación a otra.

Como desventaja principal, programar una app nativa implica que se hace única y exclusivamente para una determinada plataforma. Si se desea programar la misma app para otra plataforma, hay que hacerlo desde el principio, por lo menos en el apartado de implementación.

2.5.2. Aplicaciones web

Estas aplicaciones tienen su base en el lenguaje HTML, junto con JavaScript y CSS, herramientas muy conocidas por los programadores web.

Entre sus ventajas se encuentran las siguientes:

- No hace falta ningún SDK. Esto permite programar de forma independiente al sistema operativo, obteniendo una misma app que se puede utilizar en diferentes plataformas sin necesidad de utilizar un código diferente para cada caso particular.
- No necesitan instalarse en el dispositivo. Se visualizan utilizando el navegador web como un sitio web normal. Para facilitar su utilización se puede crear un acceso directo a la web en el escritorio del dispositivo, lo cual abriría directamente la web deseada y parecería una app más.
- No hace falta distribuirlas en las tiendas de apps. Se comercializan y promocionan de forma diferente.

- El usuario siempre está viendo la última versión, al estar accediendo a una página web.

Como inconvenientes:

- Requieren disponer de conexión a internet para funcionar correctamente.
- El acceso al hardware del dispositivo es complicado, por lo que no pueden realizar gestión de la memoria y no aprovechan al máximo la potencia de los componentes hardware.
- La interfaz suele ser genérica e independiente de la apariencia general de cada uno de los sistemas operativos donde se utiliza, con lo que el usuario requiere de un proceso previo de familiarización con los elementos de navegación e interacción.

2.5.3. Aplicaciones híbridas

Las aplicaciones híbridas son una combinación de las apps nativas y las web apps. Se desarrollan como una web app, utilizando HTML, CSS y JavaScript y una vez que la aplicación está terminada, se compila o empaqueta de modo que como resultado se obtiene una aplicación que parece nativa.

Las ventajas de este tipo de aplicaciones son:

- Casi con un mismo código se obtienen diferentes aplicaciones para diferentes plataformas.
- Pueden acceder, mediante el uso de librerías, a las capacidades hardware del dispositivo, como si se tratase de una app nativa.

Sus desventajas son:

- Su aspecto visual no se identifica con una determinada plataforma, aunque hay formas de utilizar controles y botones nativos.
- Su rendimiento es más pobre que en una app nativa.

2.6. Plataformas disponibles

Según los datos de Kantar Worldpanel [5], empresa especializada en comportamiento de los consumidores, el porcentaje de ventas de dispositivos según su sistema operativo a diciembre de 2015 se encuentra en la Ilustración 9.

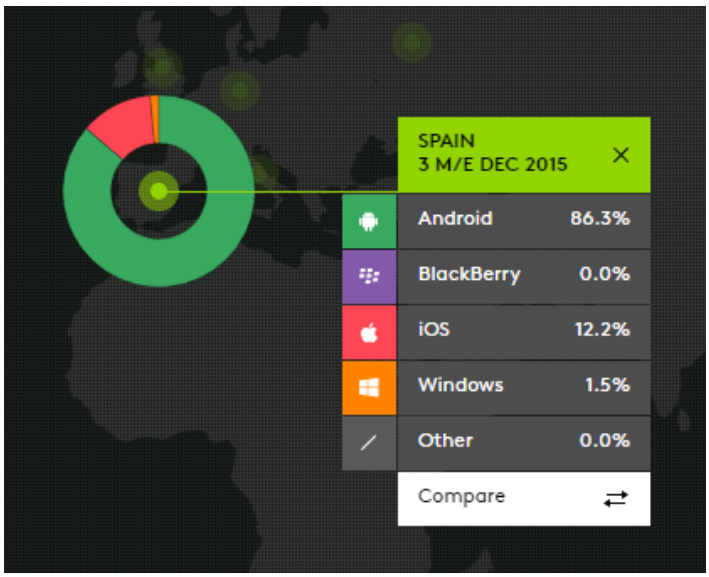


ILUSTRACIÓN 9. PORCENTAJES DE VENTAS SEGÚN SISTEMA OPERATIVO EN DICIEMBRE DE 2015

Como se puede observar, las principales plataformas actualmente son dos: Android de Google liderando las ventas con un 86,3%, e iOS de Apple con un 12,2%. Hay algunos sistemas operativos más en el mercado, como Windows Phone de Microsoft, Blackberry OS, Firefox OS de Mozilla o Ubuntu Phone de Canonical, pero su presencia es prácticamente simbólica o con ventas nulas. Recientes noticias confirman que empresas que antes ofrecían su propio sistema operativo, como era el caso de Blackberry han decidido ofrecer sus productos con el sistema operativo Android [6], han abandonado

la inclusión de su sistema operativo en móviles, como en el caso de Firefox OS [7] o están a expensas de que sus futuros proyectos provoquen un lanzamiento definitivo de su plataforma, como Microsoft y su Windows 10 Mobile, la evolución de Windows Phone [8].

La situación queda todavía más clara si se observa la Ilustración 10. En ella se comparan las cifras a diciembre de 2014 con las de diciembre de 2015 vistas en la Ilustración 9. Se puede apreciar un aumento de un 3,3% de ventas de dispositivos con Android mientras que disminuyen las de los dispositivos con iOS (-0,7%) y Windows (-2,3%). De estos datos se puede deducir que el aumento del 3,3% de Android es prácticamente la suma de pérdidas de iOS y Windows, un 3%.

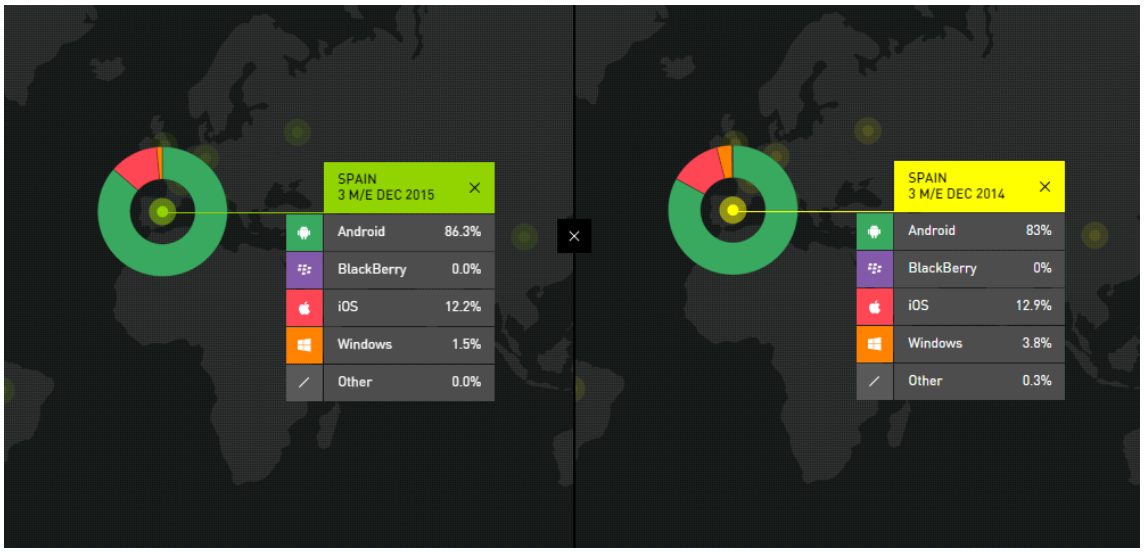


ILUSTRACIÓN 10. COMPARATIVA DE PORCENTAJES DE VENTAS ENTRE DIC. 2014 Y DIC. 2015

De esta información se puede extraer que hay un claro dominante del mercado. Los consumidores se decantan por dispositivos Android muy por encima de lo que lo hacen por dispositivos de cualquier otro sistema operativo.

Debido a lo destacado de los datos, este estudio de plataformas, que pretende utilizar el resultado como sistema operativo a utilizar para la creación de la aplicación, se va a

centrar en las dos con más ventas, dejando de lado otras opciones demasiado minoritarias como para tenerlas en cuenta para emplear dinero y recursos en ellas.

2.6.1. Android

Android es un sistema operativo basado en el núcleo Linux. Se diseñó para su utilización en dispositivos móviles con pantalla táctil. Fue desarrollado por la empresa Android Inc. hasta que en 2005 fue comprada por Google. Más tarde, en 2007, se presentó la Open Handset Alliance (OHA) [9], una alianza de empresas de telecomunicaciones y fabricantes de software y hardware, entre las que se encuentra Google, y cuya misión es el desarrollo de estándares abiertos para dispositivos móviles. La encargada de desarrollar Android para la OHA es Google, quien elabora cada versión a puerta cerrada para posteriormente liberarla en su forma AOSP (Android Open Source Project), es decir, en su versión estándar. A partir de aquí, cada fabricante toma esta versión AOSP y la modifica añadiéndole sus propias capas de personalización y los drivers de su hardware para afinar Android a sus necesidades particulares.

Al tratarse de un sistema operativo que nace del acuerdo de gran cantidad de compañías influyentes en el mercado de las telecomunicaciones, de la informática y de internet, cuenta con un apoyo gigantesco. Su código abierto permite que las compañías tomen este software y lo personalicen y esto les facilita mucho las cosas, puesto que en lugar de emplear recursos en desarrollar el sistema operativo, únicamente deben emplearlo en configurarlo según sus propios requisitos de cara a la comercialización de sus productos, a la vez que Google se centra en su desarrollo y delega la fabricación del hardware de los productos que presenta como propios en alguno de sus socios como LG, HTC, Motorola o Huawei.

Esta gran ventaja es a su vez el gran inconveniente. Muchas empresas toman la versión AOSP, la personalizan para trabajar con su hardware y lo comercializan. El posible problema viene justo después, cuando aparece una nueva versión de Android. En este punto es donde según la fortaleza de la empresa comercializadora, se emplean recursos

en personalizar la nueva versión para el antiguo hardware, cosa que realizan algunas grandes empresas como Samsung o LG durante los primeros uno o dos años, según modelos. En algunas estrategias comerciales se contempla que la versión con la que sacan el dispositivo al mercado será la única y definitiva, con lo que las nuevas versiones de Android se verán únicamente en los dispositivos más modernos, lo que de alguna manera obliga a los clientes a adquirir modelos más modernos en lugar de aprovechar el que tienen únicamente actualizando su software. En resumen, existen tantas versiones personalizadas de Android como modelos de dispositivos diferentes en el mercado. Esta libertad de personalizar es por tanto su inconveniente, puesto que si se respetase una versión estándar para todos los dispositivos igual, las actualizaciones podrían llevarse a cabo de un modo mucho más sencillo [10].

Google conoce este problema y por ello comenzó a intentar controlar un poco más su sistema operativo. No deja de ser complicado y necesario personalizar Android según el hardware de cada dispositivo, tarea que actualmente recae en los fabricantes que quieren incluir Android en sus aparatos. Estos mismos fabricantes suelen añadir algo más, una personalización que consideran oportuna. Posteriormente se dan casos de que las empresas de telecomunicaciones añaden su propia capa extra de personalización. Google, mostrando la pauta más correcta, comercializa dispositivos con su Android más reciente directo de fábrica en contados modelos que presenta como suyos, pero fabricados por otras compañías bajo sus especificaciones. Tal es el caso de sus modelos Nexus. Estos modelos son un ejemplo de cómo su software funciona en dispositivos sin capa de personalización, tal y como fue pensado que debería funcionar cuando se diseñó. Al no llevar personalización, los Nexus son los dispositivos que antes se pueden actualizar a las últimas versiones (Google garantiza que durante al menos dos años desde su salida al mercado, sus dispositivos siempre tendrán la versión Android más actualizada).

Del mismo modo, Google cambió la política de soporte de versiones y actualizaciones para los fabricantes [11]. Tal y como se puede observar en la Tabla 3, cada lanzamiento de una nueva versión tendrá una ventana de aprobación que de modo habitual se

cerrará nueve meses después de que se haya lanzado la siguiente versión de Android. En otras palabras, ningún fabricante puede certificar un dispositivo nuevo con una versión de Android que se encuentre más de dos versiones por debajo de la versión actual de Android. De este modo Google se asegura homogeneidad y modernidad en las versiones, evitando que, por ejemplo, a día de hoy, un fabricante saque al mercado un dispositivo con Android 3.x Honeycomb, cuando actualmente la versión más reciente es Android 6.0 Marshmallow, pudiendo certificar únicamente dispositivos que al menos lleven Android 4.0 Ice Cream Sandwich.

Versión del API	Fecha de lanzamiento de la versión AOSP	Fecha de cierre de aprobación
Gingerbread	06/12/2010	01/02/2014
Honeycomb	24/02/2011	01/02/2014
Ice Cream Sandwich	16/12/2011	01/02/2014
4.1 (API nivel 16)	12/07/2012	01/02/2014
4.2 (API nivel 17)	13/09/2012	24/04/2014
4.3 (API nivel 18)	25/07/2013	31/07/2014
4.4 (API nivel 19)	31/10/2013	03/10/2014

TABLA 3. CADUCIDAD DE APROBACIONES PARA LAS DIFERENTES VERSIONES DE ANDROID

Así mismo, los desarrolladores disponen de una página para desarrolladores [12] donde se muestran siempre datos actualizados sobre los dispositivos que acceden a Google Play Store, tales como versión de Android, tamaño de pantalla, etc. De modo que permite a los desarrolladores priorizar esfuerzos según el número y versión de los dispositivos Android a los que quieran dar soporte. Esto se puede observar en la Ilustración 11.

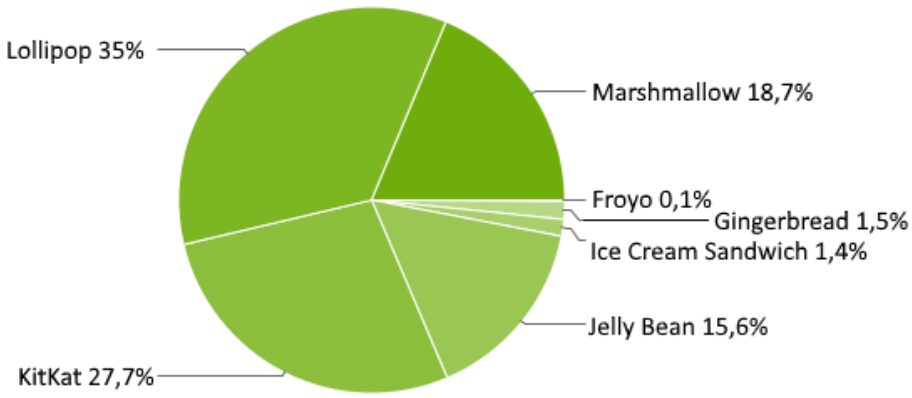


ILUSTRACIÓN 11. VERSIONES DE ANDROID UTILIZADAS EN LA ACTUALIDAD

El detalle de la distribución de las diferentes versiones mostradas en la Ilustración 11, se muestra a continuación en la Tabla 4.

Versión	Nombre en clave	API	Distribución
2.2	Froyo	8	0,1%
2.3.3 – 2.3.7	Gingerbread	10	1,5%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	1,4%
4.1.x	Jelly Bean	16	5,6%
4.2.x		17	7,7%
4.3		18	2,3%
4.4	KitKat	19	27,7%
5.0	Lollipop	21	13,1%
5.1		22	21,9%
6.0	Marshmallow	23	18,7%

TABLA 4. DISTRIBUCIÓN DE VERSIONES DE ANDROID

En cuanto al kit de desarrollo de software, se ha estado realizando durante los últimos años con el famoso IDE (Integrated Development Environment, Entorno de Desarrollo Integrado) Eclipse [13] en su versión para Java junto con el plug-in Android Development Tools (ADT) [14].

En diciembre de 2014 Google lanzó la primera versión estable de Android Studio, la 1.0, pasando a ser este el IDE oficial para el desarrollo de aplicaciones en Android, puesto que está diseñado específicamente para el desarrollo en Android, y abandonando el Eclipse ADT. Está disponible para Windows, Mac OS X y Linux. Esto proporciona muchas ventajas a los desarrolladores, que no están sujetos a la utilización de un sistema operativo concreto [15].

ILUSTRACIÓN 12. PANTALLA DE CARGA DE ANDROID STUDIO

Actualmente Android Studio ha alcanzado la versión 2.0, que incrementa drásticamente la velocidad en la que se puede construir y del flujo de trabajo [16].

A través del número de dispositivos que ejecutan la versión más reciente de Google Play Store, que es compatible con Android 2.2 y superiores, Google es capaz de conocer diversos datos estadísticos, como los datos acerca del número relativo de dispositivos que tienen una configuración de pantalla particular, definida por una combinación de

tamaño de pantalla y densidad. Para simplificar el modo en que se diseñan las interfaces de usuario, Android divide el rango de tamaños de pantalla actuales y densidades dentro de varias celdas tal y como se expresa en la Tabla 5.

Estos datos se obtienen a partir de los dispositivos que han visitado la Google Play Store en los 7 días previos. [12]

En el caso de los datos mostrados a continuación, se publicaron el 1 de febrero de 2016 y corresponden al intervalo del 25 de enero de 2016 al 31 de enero de 2016.

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Pequeño	2,4%						**2,4%**
Normal		5,1%	0,1%	41,5%	22,9%	14,8%	**84,4%**
Grande	0,3%	5,0%	2,3%	0,6%	0,5%		**8,7%**
Extra Grande		3,5%		0,3%	0,7%		**4,5%**
Total	**2,7%**	**13,6%**	**2,4%**	**42,4%**	**24,1%**	**14,8%**	

TABLA 5. TAMAÑOS DE PANTALLA Y DENSIDADES EN DISPOSITIVOS ANDROID (01/02/2016)

En la Ilustración 13 se muestra en gráfico de tarta, las distribuciones de tamaño de pantalla actuales.

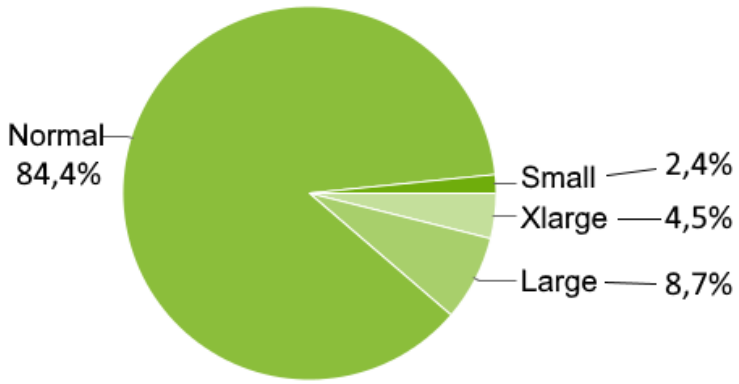


ILUSTRACIÓN 13. DISTRIBUCIÓN DE TAMAÑOS DE PANTALLA (01/02/2016)

Y en la Ilustración 14, la distribución de las diferentes densidades presentes en los dispositivos Android a febrero de 2016.

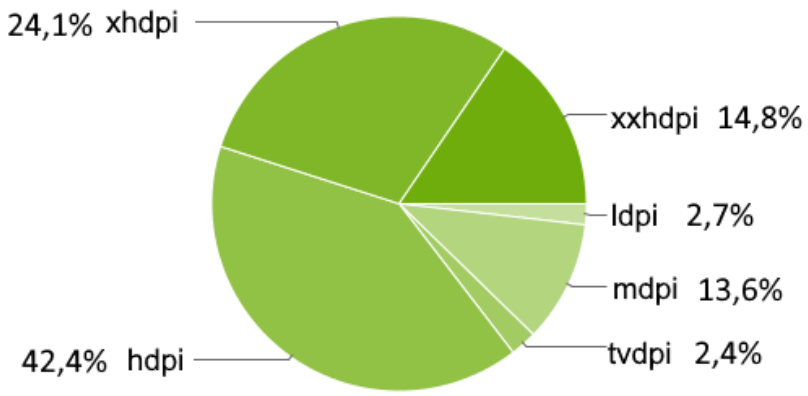


ILUSTRACIÓN 14. DISTRIBUCIÓN DE DENSIDADES DE PANTALLA (01/02/2016)

A efectos de programación, Android resulta más complejo por la cantidad de tamaños de pantalla y densidades a las que hacer frente a la hora de posicionar los objetos en pantalla, así como la disposición al voltear el dispositivo, en caso de querer permitir que cuando el usuario gire la pantalla los elementos se muestren de modo similar o acorde a la nueva forma de visualizar la pantalla.

Se pueden afrontar todas las posibilidades o limitar la utilización de la app a determinados tamaños de pantalla y densidades. Todo va a depender de la cantidad de recursos que se quieran emplear y el rango de dispositivos que se quiera abarcar.

2.6.2. iOS

iOS es un sistema operativo móvil de Apple, derivado de OS X, que a su vez está basado en Darwin BSD, por lo que se trata de un sistema operativo tipo Unix. Se desarrolló inicialmente con el nombre iPhone OS para el primer Smartphone de Apple, el iPhone, cuya primera generación salió al mercado el 29 de junio de 2007, y posteriormente para iPod touch y iPad [18].

Como iOS no permite su instalación en hardware de terceros, sólo está disponible para iPhone, iPod touch y iPad. Sucede lo mismo con OS X, que sólo está disponible para los dispositivos Apple Mac tales como Macbook, Macbook Air, Macbook Pro, iMac, Mac Pro y Mac Mini.

El IDE de desarrollo de aplicaciones para iOS, Xcode [19], así como su lenguaje de programación exclusivo, Swift [20], de los que se muestran sus respectivos logos en la Ilustración 15, se han desarrollado por Apple para completar y sustituir paulatinamente a Objective-C [21] como lenguaje de programación para las aplicaciones iOS. Tanto Xcode como Swift funcionan única y exclusivamente en plataformas Apple.

ILUSTRACIÓN 15. SWIFT Y XCODE, LAS HERRAMIENTAS OFICIALES PARA IOS

Estas restricciones suponen un inconveniente para posibles desarrolladores, puesto que deben realizar una inversión inicial en hardware exclusivo de la marca Apple, nada económico en comparación con hardware genérico válido para otras plataformas como Windows o Linux.

A día de hoy, adquirir un portátil compatible en lugar de un Mac puede suponer unos 400 o 500 euros de ahorro. Adquirir un teléfono Android de similares características a las de un iPhone supone 150 euros de ahorro, como se puede apreciar en la Ilustración 16, donde se muestran las webs oficiales del iPhone 6s y de uno de sus competidores de la misma gama de producto, el S6 de la marca Samsung. Esto sin tener en cuenta que Apple no fabrica smartphones de gama media o baja, mientras que todas las demás compañías, todas integradoras de Android en sus dispositivos, sí que fabrican productos

de gama media y baja. Es decir, adquiriendo un smartphone de gama media como por ejemplo un M5 de la marca española BQ, más que suficiente para los requisitos de la aplicación a implementar, cuyo precio en su modelo más completo es de 299,90 euros, el ahorro sería de 449,10 euros.

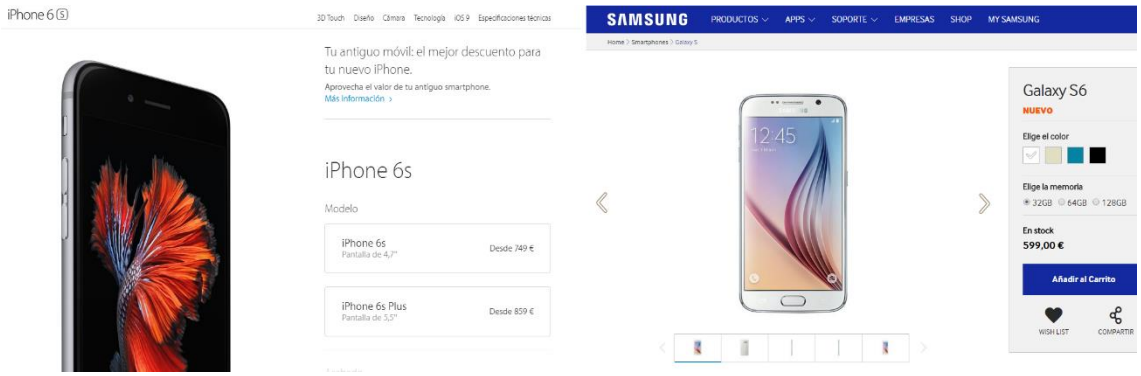


ILUSTRACIÓN 16. COMPARATIVA DE PRECIOS SEGÚN LAS WEBS OFICIALES DE APPLE Y SAMSUNG

Sin embargo, el hecho de que sólo existan 8 dispositivos iOS (iPhone 6s, iPhone 6, iPhone 5, iPad Pro, iPad Air 2, iPad Air, iPad mini 4 y iPad mini 2) facilita en gran medida el desarrollo de aplicaciones, puesto que sólo es necesario probar la app en 8 dispositivos diferentes para asegurar su compatibilidad y su correcta visualización y rendimiento. El hardware es el que hay, es conocido y no varía hasta la siguiente generación. En el caso de Android la variedad de hardware donde habría que probar la app complica este apartado.

En cuanto al soporte de versiones, Apple, como fabricante de dispositivos, mantiene el control de las versiones dejando obsoletos sus dispositivos. Desde iOS 4, al igual que Google con su política de versiones mencionada en el capítulo 2.4.1, se establecieron restricciones por las cuales se dejan fuera los dispositivos más antiguos, no permitiéndoles actualizar a versiones más recientes [22]. Por norma general, se mantienen como válidos dispositivos de hasta 2 generaciones. Es decir, a fecha actual, la versión más moderna del iPhone es la 6S, con sistema operativo iOS 9, mientras que el iPhone 4 queda fuera, al no poderse actualizar por encima de iOS 7. Del mismo modo, Apple obliga a utilizar un SDK que compile en versiones para iOS 8, por lo que las apps

de la App Store sólo pueden instalarse en dispositivos con iOS 8 o superior. De este modo se obliga a los consumidores a adquirir modelos más modernos de iPhone para seguir recibiendo actualizaciones de las apps que utilizan y que se continúan actualizando bajo nuevos requisitos mínimos de versión de iOS.

2.6.3. Resumen comparativo de plataformas

Los dos sistemas operativos para smartphones del momento son potentes y versátiles. Android está más extendido en muchos más dispositivos, lo cual es a su vez un inconveniente a la hora de probar las aplicaciones creadas. iOS no está tan extendido, principalmente porque es exclusivo de una gama pequeña de productos hardware, que a su vez es exclusiva también en cuanto al precio. Esto mismo es su ventaja, puesto que al haber un número contado de dispositivos diferentes, es más fácil probar la app y depurarla para que su rendimiento sea el correcto en todos.

En la Tabla 6 se comparan las características principales de las dos plataformas.

	Android	**iOS**
Multiplataforma para desarrollo	✓	✗
Mayor número de dispositivos	✓	✗
Facilidad de pruebas	✗	✓
Rango de precios	✓	✗

TABLA 6. COMPARATIVA ANDROID - IOS

Como se puede observar en la **¡Error! No se encuentra el origen de la referencia.**, el sistema operativo Android parece el candidato a ser la plataforma a utilizar, pero el siguiente apartado, que revisa el perfil de cliente, será el que termine de identificar cuál es la más adecuada.

2.7. Perfil del cliente

Como todo producto que pretende salir al mercado, hay que determinar a qué clientes va dirigido. Es lo que se conoce con el nombre de perfil del cliente. Definirlo ayuda en gran medida a valorar sus posibilidades de éxito.

Esta aplicación nace con la intención de ayudar a llevar un registro de dispositivos, de su historial de intervenciones, ficha completa de características, etc. Es decir, está pensada para su utilización por personal técnico que en un determinado momento visitará las instalaciones de sus clientes, leerá los códigos de barras de los dispositivos y obtendrá información relevante para su trabajo.

De este modo, podemos definir un perfil de cliente ideal: pequeñas o medianas empresas con trabajadores que se desplazan a sus clientes, con teléfono de gama media o superior propio o suministrado por la empresa.

2.8. Oportunidades de negocio

Para concluir, se necesita saber la distribución de ingresos por aplicaciones. ¿Cuántas empresas de aplicaciones son sostenibles? Según la **¡Error! No se encuentra el origen de la referencia.**, con los datos extraídos de Developer Economics: State of Nation Q3 2014 [22], en general, resulta más rentable implementar en iOS frente a Android. En todas las franjas de ingresos iOS supera a Android, siendo especialmente llamativo que casi la mitad de las aplicaciones en Android reciben ingresos menores a 100$.

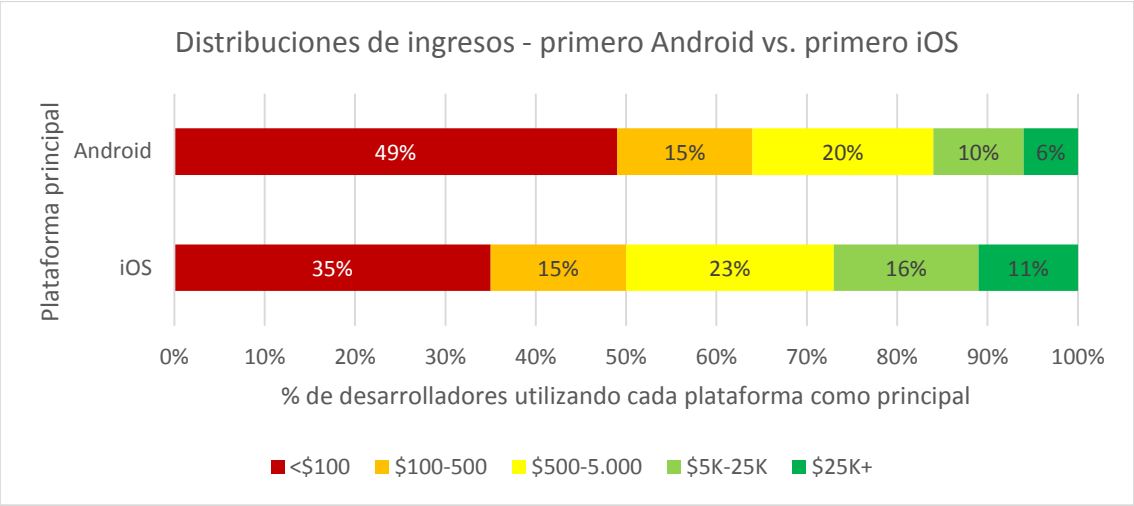


ILUSTRACIÓN 17. DISTRIBUCIÓN DE INGRESOS SEGÚN LA PLATAFORMA UTILIZADA

Sin embargo, el mismo estudio, la **¡Error! No se encuentra el origen de la referencia.** desvela que hay una oportunidad de negocio sin explotar en las aplicaciones para empresa, que es exactamente al sector al que se va a dirigir la aplicación.

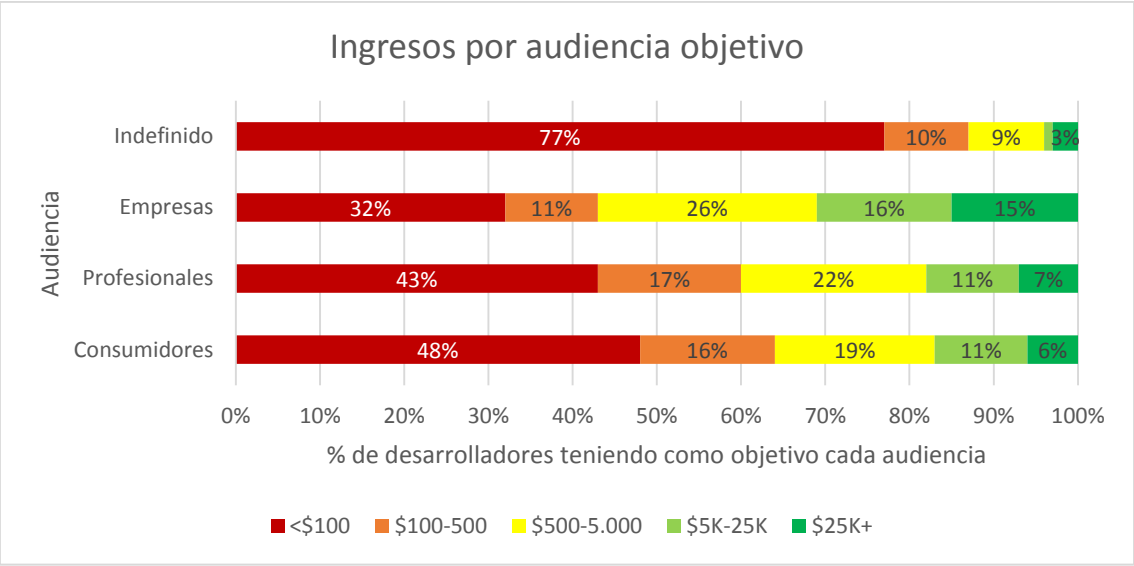


ILUSTRACIÓN 18. INGRESOS POR AUDIENCIA OBJETIVO

Dicho estudio explica además, mostrado en la Ilustración 19, que 2 de cada 3 desarrolladores tienen como objetivo a los consumidores cuando tener de objetivo a las

empresas es mucho más lucrativo, puesto que las empresas pagan mucho más por el software que los consumidores.

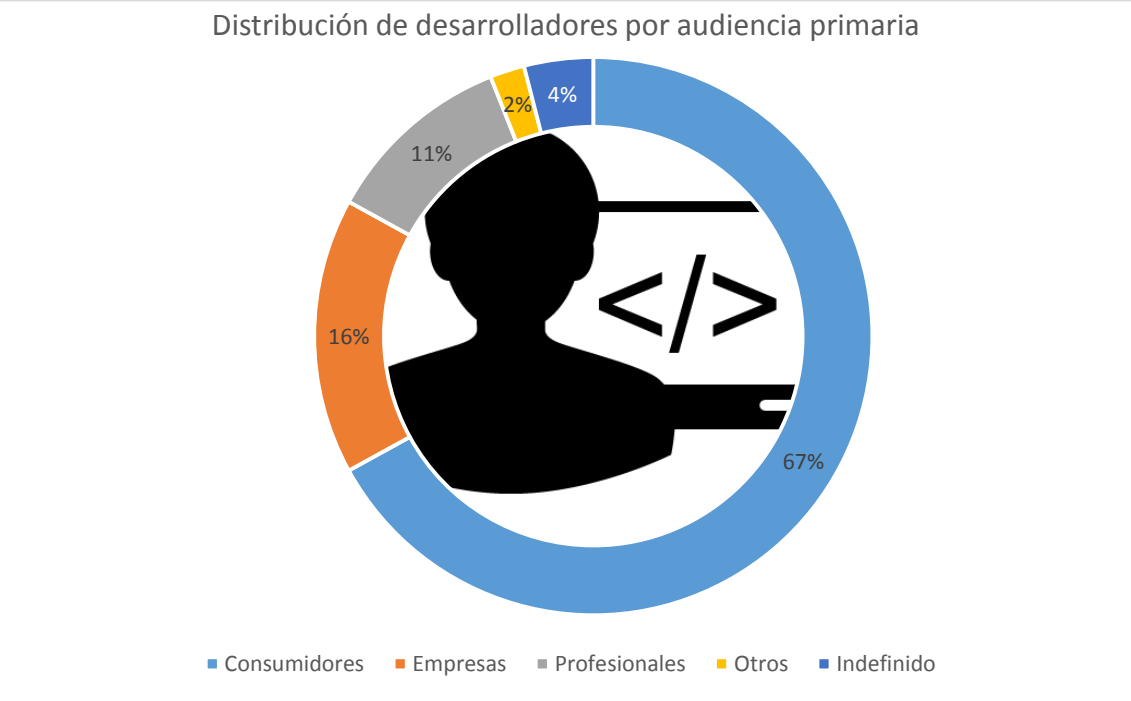


ILUSTRACIÓN 19. DISTRIBUCIÓN DE DESARROLLADORES POR AUDIENCIA OBJETIVO

Como dato adicional, la Ilustración 20, detalla qué plataformas prefieren los desarrolladores de empresa.

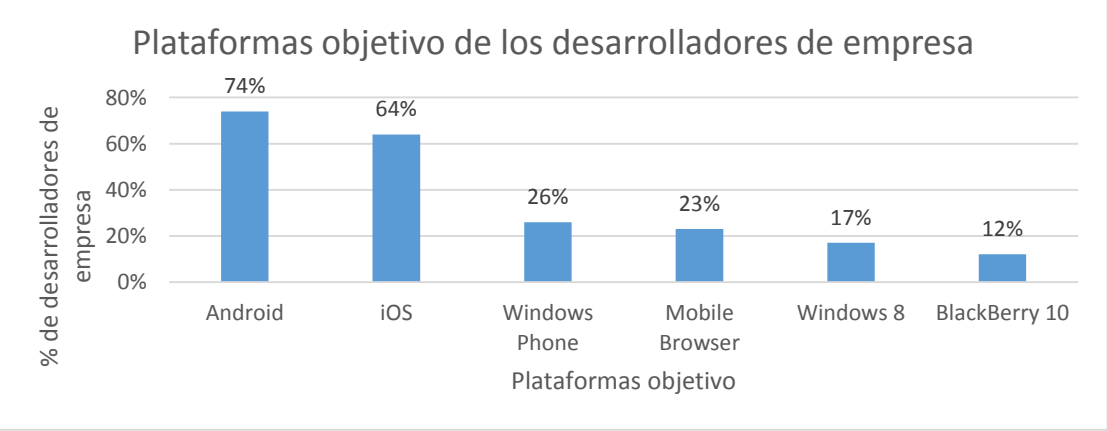


ILUSTRACIÓN 20. PLATAFORMAS PREFERIDAS POR LOS DESARROLLADORES DE EMPRESA

Con todos estos datos, se puede completar la Tabla 6 del apartado anterior, elaborando la Tabla 7 con la información adicional de este apartado.

	Android	iOS
Multiplataforma para desarrollo		
Mayor número de dispositivos		
Facilidad de pruebas		
Rango de precios		
Mayor rentabilidad general		
Mayor rentabilidad empresas		

TABLA 7. CONCLUSIONES FINALES ANDROID - IOS

En resumen, si la aplicación tiene como objetivo a trabajadores, debe decantarse por Android, dado que si hay más usuarios con Android que con iOS, el subconjunto de trabajadores con Android es mayor que el de trabajadores con iOS. Y como confirma el estudio de Developers Economics, realizar una aplicación empresarial tiene más probabilidades de recibir ingresos que una aplicación para consumidores. Por lo tanto, todo indica que la elección adecuada es comenzar por Android para este proyecto en particular.

Capítulo 3. Gestión de Proyecto

La gestión de proyecto sirve para describir las tareas realizadas en este proyecto y su planificación en tiempo y en asignación de recursos utilizados para su desarrollo, así como la estimación de los costes derivados de todo ello.

3.1. Organización del trabajo

La organización del trabajo consiste en detallar las tareas, asignar los recursos y especificar los productos que se obtienen como resultado.

3.1.1. Organización de las tareas WBS

La Estructura de Descomposición del Trabajo, EDT o WBS por sus siglas en inglés (*Work Breakdown Structure*), se utiliza para detallar el proyecto en unidades más concretas y descomponer, a su vez, estas unidades en componentes. De este modo se obtiene una presentación simple y organizada del trabajo requerido para completar el proyecto [23]. El WBS de este proyecto se muestra en la Ilustración 21.

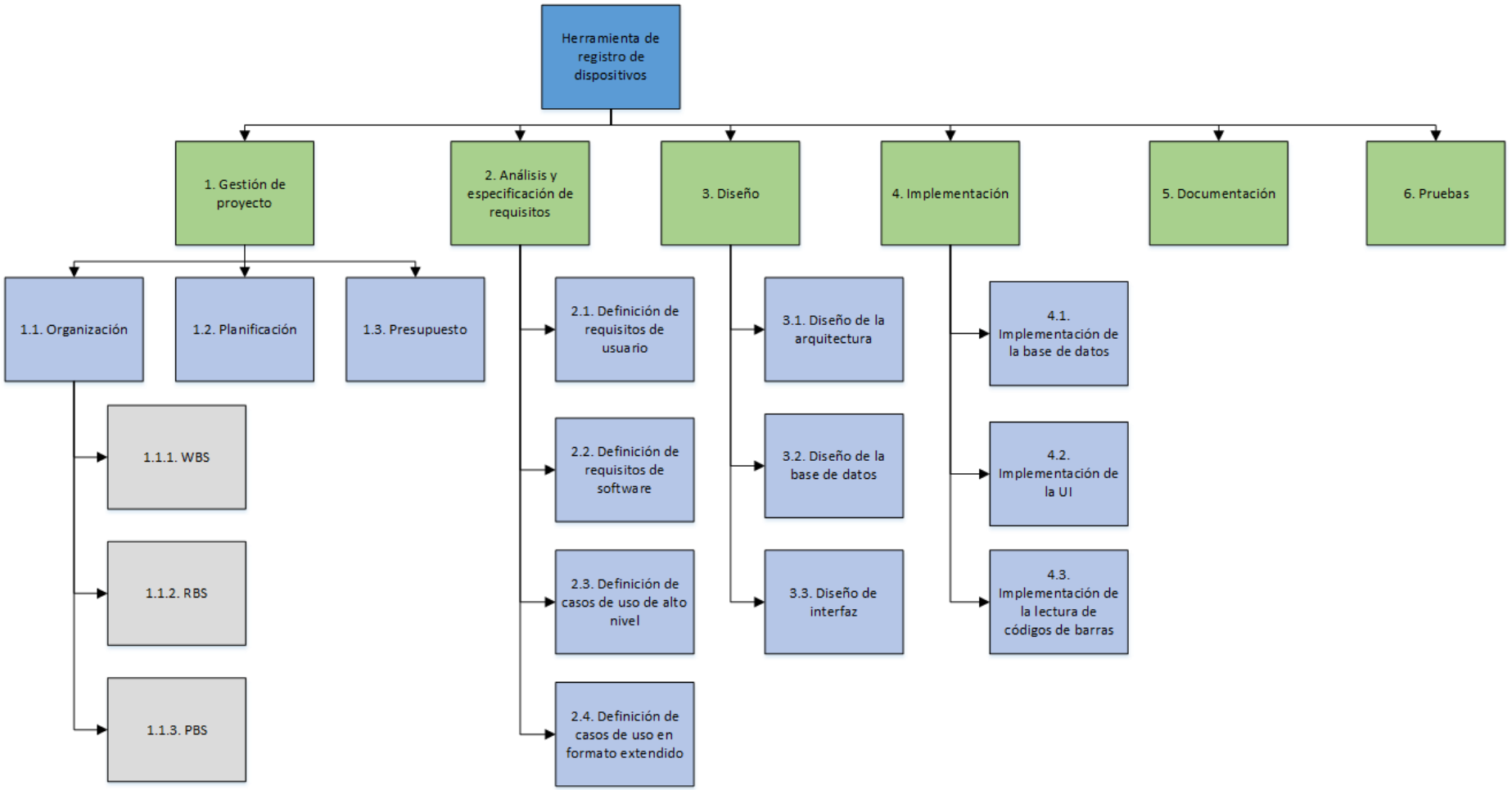


ILUSTRACIÓN 21. DIAGRAMA WBS

3.1.2. Organización de recursos RBS

La estructura detallada de recursos, más conocida por sus siglas en inglés RBS (*Resource Breakdown Structure*) es un diagrama de recursos relacionados jerárquicamente y que facilita la planificación y el control de los recursos en el proyecto. La Ilustración 22 muestra el diagrama correspondiente a este proyecto.

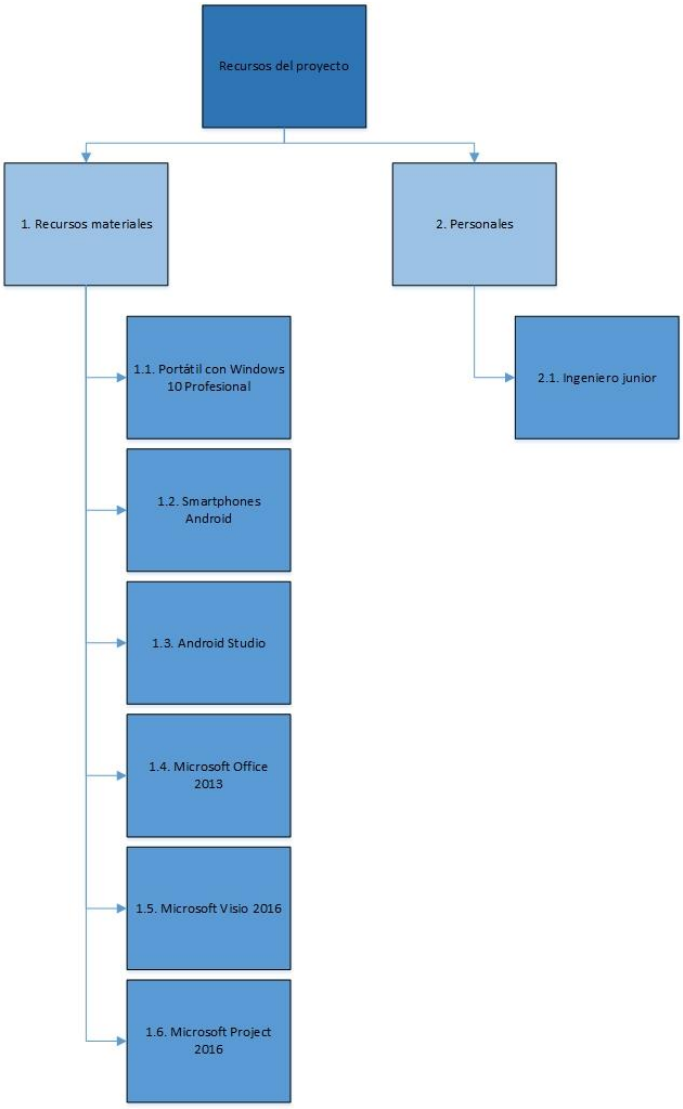


ILUSTRACIÓN 22. DIAGRAMA RBS

3.1.3. Organización de los productos PBS

La estructura de producto detallada, PBS por sus siglas en inglés (*Product Breakdown Structure*), es una herramienta para analizar, documentar y comunicar los productos de salida de un proyecto, organizados en una relación parte-todo. [24]. Este diagrama ayuda a construir una estructura de la división del trabajo. Es idéntico en formato al WBS pero incluye los productos creados como resultado en cada etapa.

En la Ilustración 23 se muestra el diagrama PBS correspondiente a este proyecto.

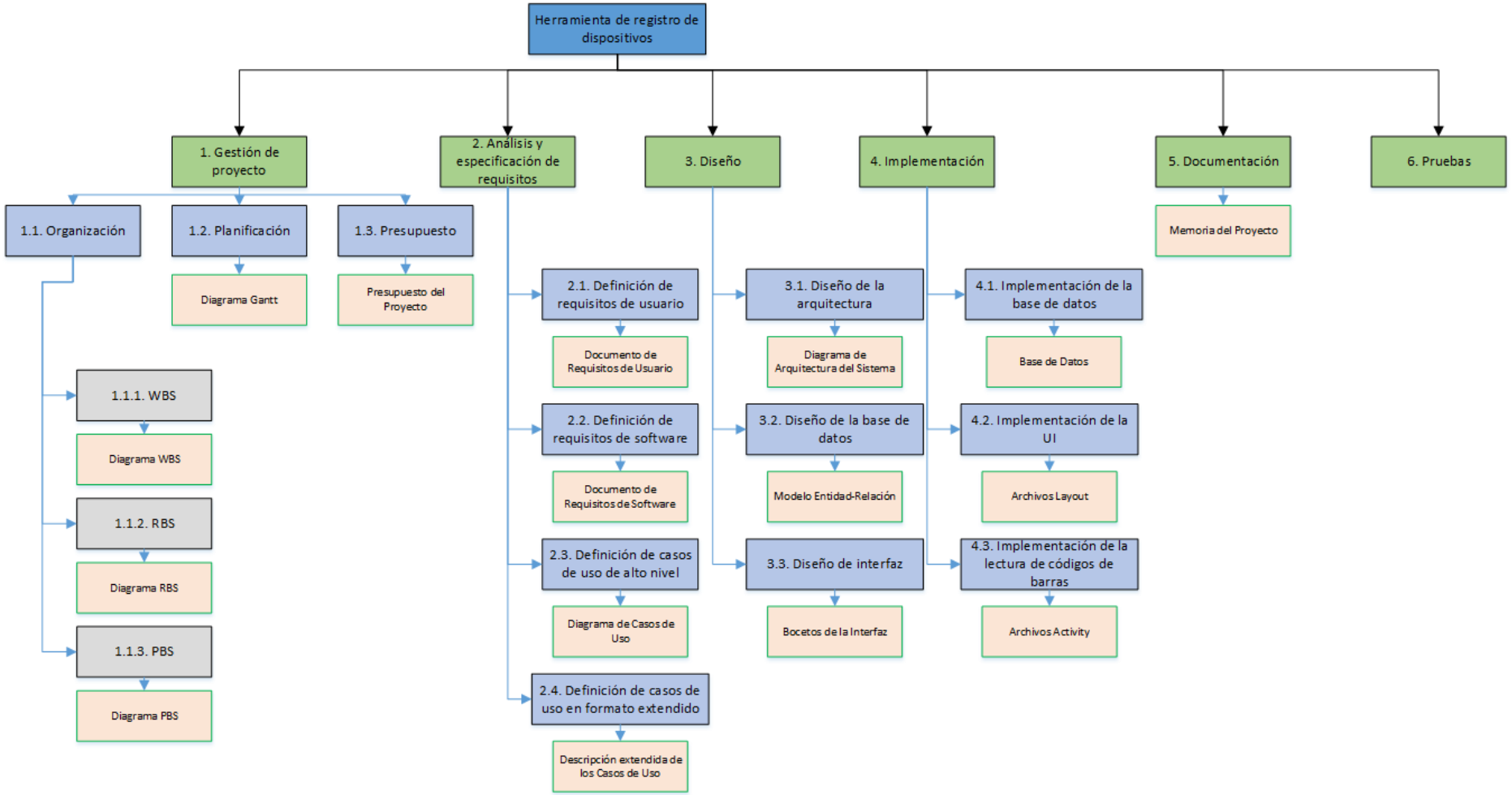


ILUSTRACIÓN 23. DIAGRAMA PBS DEL PROYECTO

3.2. Planificación

La planificación consiste en mostrar las tareas que se realizarán a lo largo del proyecto así como la duración de las mismas, a modo de cronograma. Se pueden producir variaciones en función del cumplimiento de las tareas en el tiempo previsto o no, de modo que se pueden producir cambios en dicho cronograma que, mediante la modificación de la tarea retrasada o adelantada, podría modificar la fecha estimada final. Esto proporciona una valiosa estimación rápida de un posible retraso o adelanto tanto de cara al cliente como al equipo que lleva a cabo el proyecto.

Toda la planificación se ha basado en la suposición de un único ingeniero trabajando 40 horas a la semana 100% en el proyecto, de lunes a viernes. En el calendario se han tenido en cuenta los siguientes días festivos:

- 24 de marzo: Jueves Santo.
- 25 de marzo: Viernes Santo.
- 2 de mayo: Fiesta del Dos de mayo (fiesta local en Madrid)
- 16 de mayo: Fiesta de San Isidro (fiesta local en Madrid), desplazada del domingo al lunes.

En la Ilustración 24 se muestra el detalle de tareas a llevar a cabo.

DIAGRAMA DE GANTT

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**App de registro de dispositivos**	**75 días**	**sáb 13/02/16**	**jue 26/05/16**	
2		**Análisis Inicial**	**10 días**	**sáb 13/02/16**	**jue 25/02/16**	
3		Análisis de la situación actual	10 días	sáb 13/02/16	jue 25/02/16	
4		**Gestión de Proyecto**	**6 días**	**vie 26/02/16**	**vie 04/03/16**	**3**
5		Organización	2 días	vie 26/02/16	lun 29/02/16	3
6		Planificación	2 días	mar 01/03/16	mié 02/03/16	5
7		Presupuesto	2 días	jue 03/03/16	vie 04/03/16	6
8		**Análisis y especificación de requisitos**	**10 días**	**lun 07/03/16**	**vie 18/03/16**	**4**
9		Definición de requisitos de usuario	2 días	lun 07/03/16	mar 08/03/16	7
10		Definición de requisitos de software	2 días	mié 09/03/16	jue 10/03/16	9
11		Definición de casos de uso de alto nivel	3 días	vie 11/03/16	mar 15/03/16	10
12		Definición de casos de uso en formato extendido	3 días	mié 16/03/16	vie 18/03/16	11
13		**Diseño**	**12 días**	**lun 21/03/16**	**mar 05/04/16**	**12**
14		Diseño de la arquitectura	2 días	lun 21/03/16	mar 22/03/16	12
15		Diseño de la base de datos	3 días	mié 23/03/16	vie 25/03/16	14
16		Diseño de la interfaz	10 días	mié 23/03/16	mar 05/04/16	14
17		**Implementación**	**15 días**	**mié 06/04/16**	**mar 26/04/16**	**13**
18		Implementación de la base de datos	3 días	mié 06/04/16	vie 08/04/16	15
19		Implementación de la UI	5 días	mié 06/04/16	mar 12/04/16	16
20		Implementación de la lectura de códigos de barras	10 días	mié 13/04/16	mar 26/04/16	19
21		**Pruebas**	**2 días**	**mié 27/04/16**	**jue 28/04/16**	**17**
22		Pruebas	2 días	mié 27/04/16	jue 28/04/16	
23		**Documentación**	**20 días**	**vie 29/04/16**	**jue 26/05/16**	**21**
24		Memoria del Proyecto	20 días	vie 29/04/16	jue 26/05/16	21

DIAGRAMA DE GANTT

ILUSTRACIÓN 24. TAREAS DEL PROYECTO

En la Ilustración 25 se muestra el diagrama Gantt resumido con las fechas estimadas de finalización de cada una de las tareas resumen.

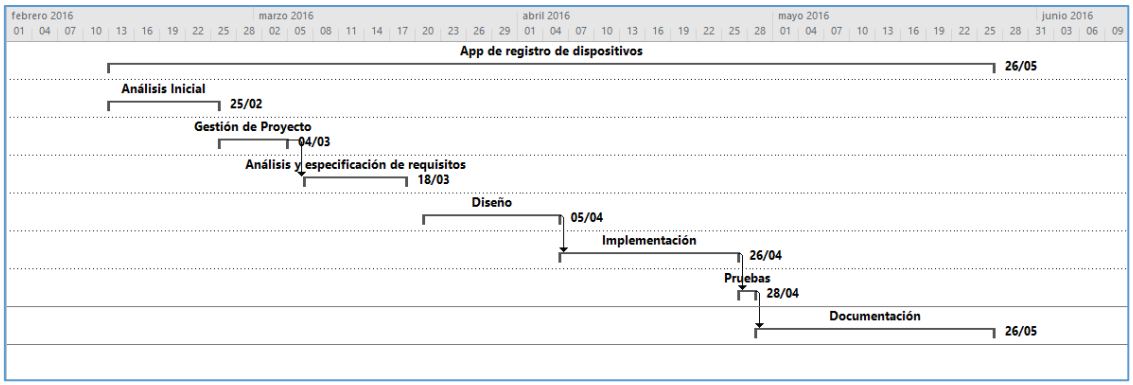


ILUSTRACIÓN 25. DIAGRAMA GANTT DEL PROYECTO

3.3. Presupuesto

Tras establecer la duración y planificación del proyecto, se presenta el presupuesto para su realización. Este se compone de recursos humanos y de costes materiales.

3.3.1. Recursos humanos

Se considera suficiente un único ingeniero dedicado por completo a la realización del proyecto. En la Tabla 8 se muestra el coste total de dicho ingeniero trabajando a tiempo completo en el proyecto.

Nombre	Comienzo	Fin	Tiempo	Precio/hora	Coste
Ingeniero	15/02/16	26/05/16	600 horas	15€	9.000€

TABLA 8. COSTES DE LOS RECURSOS HUMANOS

3.3.2. Costes materiales

Para la realización del proyecto se requieren ciertos materiales sin los cuales no es viable su desarrollo e implementación. Estos materiales se encuentran detallados a continuación en la Tabla 9.

Nombre del recurso	Coste
Microsoft Office 2016	539,00€
Microsoft Visio 2016	399,00€
Microsoft Project 2016	769,00€
Ordenador con Windows 10	1.500,00€
Smartphone BQ	299,00€
Smartphone Samsung	100,00€
Total	**3.606,00€**

TABLA 9. COSTES MATERIALES

Hay que tener en cuenta que estos gastos iniciales se deben al tratarse del primer proyecto, como inversión inicial. Estos gastos, por sus características, podrían, y deben, ser amortizados a lo largo del tiempo, durante, aproximadamente 2 años, puesto que se pueden emplear en proyectos posteriores.

No se han incluido en los costes otros materiales utilizados al tratarse de herramientas gratuitas descargables de internet, tales como Android Studio de Google [25], el entorno de desarrollo Java de Oracle [26], el entorno de desarrollo Eclipse [27] o el software de tratamiento de imágenes Paint.Net de Rick Brewster [28]

3.3.3. Resumen de costes

Vistos los costes en recursos humanos y en costes materiales, en la Tabla 10 se muestra un resumen de los costes estimados totales del proyecto. En dichos costes se han incluido dos porcentajes: el riesgo, para amortiguar posibles gastos adicionales por retrasos o imprevistos y el beneficio o plusvalía que se pretende obtener del producto software en el mercado.

Este resumen de costes es una simplificación que pretende mostrar el coste del proyecto en sí y se dejan fuera, intencionadamente, por encontrarse fuera del ámbito de este documento, otros gastos fijos y variables que forman parte de este y cualquier otro

proyecto que se quisiera llevar a cabo: luz, agua, teléfono, conexión a internet, alquiler del local u oficina, limpieza, impuestos asociados, desplazamientos, etc.

Recurso	Coste
Costes de RRHH	9.000,00€
Costes materiales	3.606,00€
Total	12.606,00€
Riesgo (15%)	1.890,90€
Beneficio (15%)	1.890,90€
Precio total	16.387,80€
Precio total IVA incluido (21%)	**19.829,24€**

TABLA 10. RESUMEN DE COSTES

Capítulo 4. Desarrollo

4.1. Introducción

En este capítulo se va a realizar a cabo la fase de análisis, diseño e implementación de la aplicación.

En el análisis se estudiarán los requisitos de usuario de la aplicación, así como los casos de uso que se pueden producir. Posteriormente, en la fase de diseño del proyecto se detallará la arquitectura utilizada. En último lugar se comentarán los aspectos más importantes de la implementación de la aplicación.

4.2. Análisis

En la fase de análisis se especifican los requisitos de usuario, dando lugar al documento de Especificación de Requisitos Software (ERS) y sus correspondientes casos de uso.

En este apartado se llevarán a cabo las siguientes actividades:

1. Definición de los requisitos de software
2. Diagrama de Casos de Uso
3. Definición de los Casos de Uso de alto nivel
4. Definición de los Casos de Uso en formato expandido
5. Creación de la matriz de trazabilidad de los requisitos
6. Implementación de un prototipo

4.2.1. Requisitos Software

Utilizando el documento *Especificación de Requisitos según el estándar de IEEE 830-1998* [29], se detallan a continuación los requisitos mediante tablas en las que se incluyen los siguientes datos:

- Identificador: codificación que identifica de manera única un requisito. Esta codificación seguirá el siguiente patrón:
    - RF_**: Requisito funcional.
    - RNF_**: Requisito no funcional.
        - Donde ** corresponde al número de requisito.
- Necesidad: los requisitos podrán ser: esenciales, recomendables y optativos, según su importancia.
- Descripción: explicación breve del requisito de usuario.
- Fuente: entidad de la que se ha obtenido el requisito de usuario.
- Verificabilidad: capacidad del requisito para comprobar que el sistema lo cumple. Para comprobarlo:
    - El requisito debe estar incorporado en el diseño.
    - Hay que probar que el software implementa el requisito.
    - Hay que probar el requisito una vez implementado en el software.
- Dependencias: se especifica si el requisito de usuario depende de otro.

4.2.1.1. Requisitos Funcionales

Los requisitos funcionales definen el funcionamiento interno del sistema. Deben definir:

- Qué debe hacer un requisito.
- El proceso de transformación de las entradas de información en salidas de resultados.

Identificador	RF_01. Escanear códigos
Necesidad	Esencial
Descripción	El sistema leerá códigos de barras mediante la aplicación.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RF_02. Consultar registros
Necesidad	Esencial
Descripción	El sistema podrá consultar los datos previamente introducidos en el sistema.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RF_03. Alta de registros
Necesidad	Esencial
Descripción	El sistema realizará altas de registros. Los datos que se insertarán son: - Código de barras. - Nombre. - Especificaciones. - Notas adicionales.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RF_04. Modificar registros
Necesidad	Esencial
Descripción	El sistema modificará datos existentes de un registro.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RF_05. Borrar registros
Necesidad	Esencial
Descripción	El sistema eliminará registros existentes.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RF_06. Introducir códigos manualmente
Necesidad	Recomendable
Descripción	El sistema insertará códigos manualmente de forma alternativa a la lectura mediante la cámara.
Fuente	Cliente
Verificabilidad	Alta

4.2.2.2. Requisitos No Funcionales

Los requisitos no funcionales, también llamados atributos de calidad, definen requisitos que especifican criterios que pueden utilizarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales. Es decir, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino a características de funcionamiento, es decir, el cómo debe comportarse el sistema en lugar de qué debe hacer el sistema.

Identificador	RNF_01. Sistema operativo
Necesidad	Esencial
Descripción	El sistema se implementará de modo que funcione en Android 4 o superior.
Fuente	Resultado del Estudio de viabilidad de la plataforma a utilizar.
Verificabilidad	Alta

Identificador	RNF_02. Permisos de la aplicación
Necesidad	Esencial
Descripción	El sistema solicitará los siguientes permisos para el correcto funcionamiento de la aplicación: - Acceso a la cámara. - Acceso a la SD.
Fuente	Desarrollador
Verificabilidad	Alta

Identificador	RNF_03. Idioma de la interfaz
Necesidad	Recomendable
Descripción	La interfaz se mostrará en inglés.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RNF_04. Rendimiento
Necesidad	Esencial
Descripción	Se debe diseñar el sistema de forma que se obtenga el mejor rendimiento.
Fuente	Cliente
Verificabilidad	Baja

Identificador	RNF_05. Instalación simple
Necesidad	Recomendable
Descripción	El sistema debe funcionar por sí mismo, sin necesidad de instalar programas adicionales.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RNF_06. Hardware necesario
Necesidad	Recomendable
Descripción	Para un correcto funcionamiento del lector de códigos de barras es necesario que la cámara del dispositivo tenga función de autofoco.
Fuente	Desarrollador
Verificabilidad	Alta

Identificador	RNF_07. Base de datos
Necesidad	Esencial
Descripción	El sistema dispondrá de una base de datos con la información necesaria.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RNF_08. Escaneo de códigos
Necesidad	Esencial
Descripción	Se escanearán los códigos de barras mediante la cámara del dispositivo.
Fuente	Cliente
Verificabilidad	Alta

Identificador	RNF_09. Sistema
Necesidad	Esencial
Descripción	El sistema en el que se implementará la aplicación deberá ser Android 4 o superior.
Fuente	Resultado del Estudio de viabilidad de la plataforma a utilizar
Verificabilidad	Alta

Identificador	RNF_10. Permisos de la aplicación
Necesidad	Esencial
Descripción	Cuando se instala la aplicación el usuario tendrá que conceder los siguientes permisos para el correcto funcionamiento de la aplicación: - Acceso a la cámara. - Acceso a la SD.
Fuente	Desarrollador
Verificabilidad	Alta

4.2.4. Casos de uso

En este apartado se incluyen los distintos casos de uso planteados para la aplicación. Los casos de uso son documentos que describen una serie de eventos que un actor (un agente externo) lleva a cabo en un sistema para completar un proceso. Para ello se usarán diagramas de casos de uso, en los que se muestra al usuario como actor del caso interaccionando con las diferentes actividades.

4.2.4.1. Actores

Debido a la sencillez del sistema inicial, sólo es necesario un actor. Este será el propio usuario del sistema, de ahora en adelante “Usuario”.

4.2.4.2. Diagrama de casos de uso

En la Ilustración 26 se muestra el diagrama de casos de uso, representando la forma en la que el actor, en este caso el usuario, opera con el sistema.

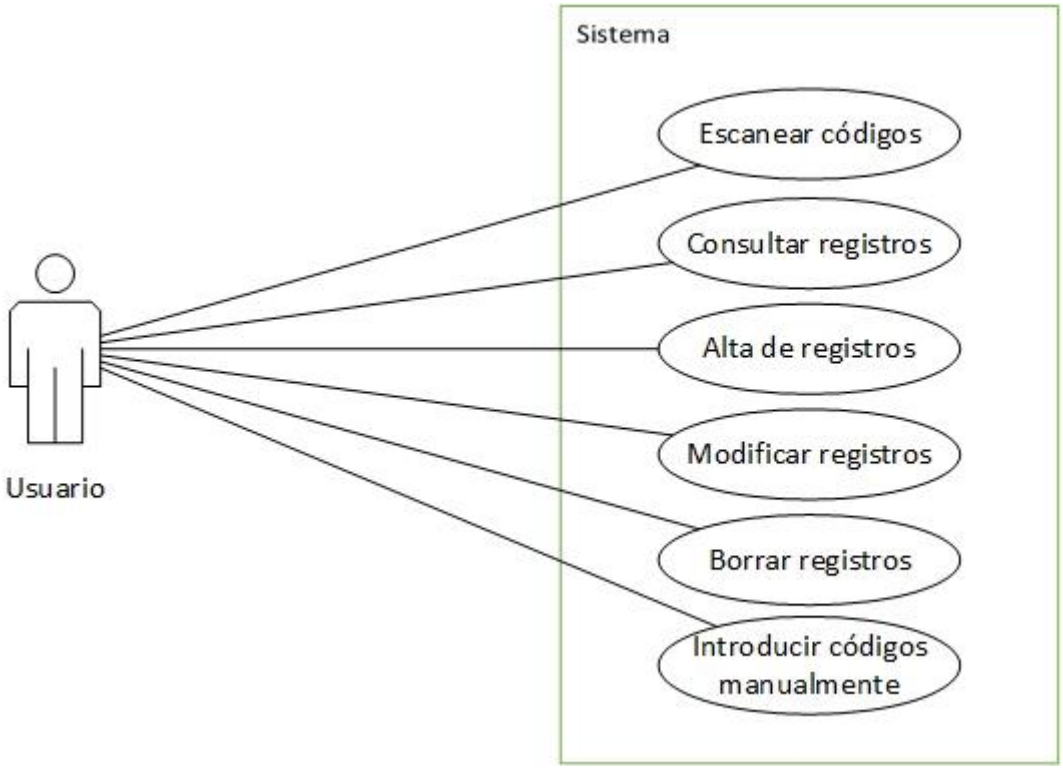


ILUSTRACIÓN 26. DIAGRAMA DE CASOS DE USO

Adicionalmente se incluye una descripción en formato tabular en la que se describen los pasos necesarios para los distintos escenarios planteados, conteniendo los siguientes pasos:

- Identificador: identifica de manera única al caso de uso. Sigue la nomenclatura CU_**, donde ** corresponde es el número de caso de uso.
- Actor: agente que interactúa con el sistema en el caso de uso.
- Objetivo: breve explicación del caso de uso.
- Precondición: condición que debe cumplirse como requisito previo al caso de uso.
- Escenario: explicación detallada de los pasos que realiza el usuario.
- Postcondición: condición resultado de la ejecución del caso de uso.

Identificador	CU_01. Escanear código
Actor	Usuario
Objetivo	Leer códigos de barras utilizando la cámara
Precondición	- La cámara no debe estar en uso por otra aplicación. - El código de barras debe ser legible.
Escenario	1. Seleccionar la opción de “Escanear” 2. Insertar el código de barras. 3. Comprobar si la inserción se ha realizado correctamente. 4. Comprobar si el registro ya está insertado, en cuyo caso se mostrará la información del artículo. En caso contrario se insertará la información. 5. Salir.
Postcondición	Aparece la ficha del artículo existente o bien una ficha en blanco para rellenar.

Identificador	CU_02. Consultar registro
Actor	Usuario
Objetivo	Leer códigos de barras utilizando la cámara
Precondición	- Haber escaneado un código. - El código tiene que existir previamente.
Escenario	1. Seleccionar la opción de “Escanear” 2. Insertar el código de barras. 3. Comprobar si la inserción se ha realizado correctamente. 4. Se muestra una ficha con el registro correspondiente al código. 5. Salir.
Postcondición	Aparece la ficha del artículo existente.

Identificador	CU_03. Alta de registro
Actor	Usuario
Objetivo	Leer códigos de barras utilizando la cámara
Precondición	- La cámara no debe estar en uso por otra aplicación. - El código de barras debe ser legible. - El registro no existe en la base de datos.
Escenario	1. Seleccionar la opción de “Escanear” 2. Insertar el código de barras. 3. Comprobar si la inserción se ha realizado correctamente. 4. Se muestra una ficha en blanco, con el código nuevo. 5. Insertar la información deseada. 6. Seleccionar opción “Guardar”. 7. Salir.
Postcondición	Se crea un registro con la información introducida.

Identificador	CU_04. Modificar un artículo
Actor	Usuario
Objetivo	Modificar datos de un artículo de la base de datos
Precondición	- El artículo tiene que haber sido introducido previamente
Escenario	1. Seleccionar la opción de “Escanear” 2. Insertar el código de barras. 3. Comprobar si la inserción se ha realizado correctamente. 4. Se muestra una ficha con el registro correspondiente al código. 5. Modificar la información deseada. 6. Seleccionar opción “Guardar”. 7. Salir.
Postcondición	Se modifica la información del registro.

Identificador	CU_05. Borrar un artículo
Actor	Usuario
Objetivo	Eliminar un artículo existente
Precondición	El artículo tiene que haber sido grabado en la base de datos previamente.
Escenario	1. Seleccionar la opción de “Escanear” 2. Insertar el código de barras. 3. Comprobar si la inserción se ha realizado correctamente. 4. Se muestra una ficha con el registro correspondiente al código. 5. Seleccionar la opción “Borrar”. 6. Aparece un mensaje pidiendo confirmación de si realmente se desea eliminar el registro. Si se responde sí, se borra. Si se responde no, no se elimina nada. 7. Salir.
Postcondición	Se elimina el artículo de la base de datos.

Identificador	CU_06. Introducir código de barras
Actor	Usuario
Objetivo	Introducir códigos de barras utilizando el teclado.
Precondición	
Escenario	1. Elegir la opción “Introducir código manualmente”. 2. Se introduce el código de barras manualmente. 3. Seleccionar la opción “Introducir”. 4. Comprobar si el registro ya está insertado, en cuyo caso se mostrará la información del artículo. En caso contrario se insertará la información. 5. Salir.
Postcondición	Aparece la ficha del artículo existente o bien una ficha en blanco para rellenar.

4.2.5. Matriz de trazabilidad

A modo de resumen, mostrando más claramente qué requisito de software cumple qué caso de uso, la Tabla 11 muestra la matriz de trazabilidad.

Requisito de Software	Casos de uso
RF_01	CU_01
RF_02	CU_02
RF_03	CU_03
RF_04	CU_04
RF_05	CU_05
RF_06	CU_06

TABLA 11. MATRIZ DE TRAZABILIDAD

4.3. Diseño

Tras haber realizado un análisis de los requisitos y especificar las funcionalidades de la aplicación, hay que definir su diseño dentro del ciclo de desarrollo de software. Para ello se describe a continuación la arquitectura de la aplicación, el diseño de la base de datos y se realizarán comentarios sobre los diferentes diseños para la interfaz de usuario.

4.3.1. Diseño de la arquitectura

En este apartado se va a analizar la arquitectura que debe tener la aplicación, así como un estudio de los componentes que forman parte de ella. Esto permite trazar los requisitos descritos en la fase de análisis de modo que se puede verificar que la implementación realizada cumple con las funcionalidades descritas en el apartado de análisis.

Como casi todas las aplicaciones móviles, la aplicación tendrá un diseño basado en Modelo-Vista-Controlador (MVC), de modo que se separan los datos de la interfaz de usuario y de la lógica de control. Con ello se consigue una mayor transparencia y se mejora la depuración, al tiempo que se optimiza la reutilización de componentes y, por lo tanto, se facilita el desarrollo de esta y posteriores aplicaciones.

Para diseñar un diagrama de componentes y establecer sus diferentes elementos, es necesario hacer una división de alto nivel de la aplicación. En este caso, se considera que se debe dividir la aplicación en dos grandes componentes: el módulo de lectura mediante la cámara o con introducción manual y la base de datos donde se almacenará la información y sobre la que se realizarán las consultas.

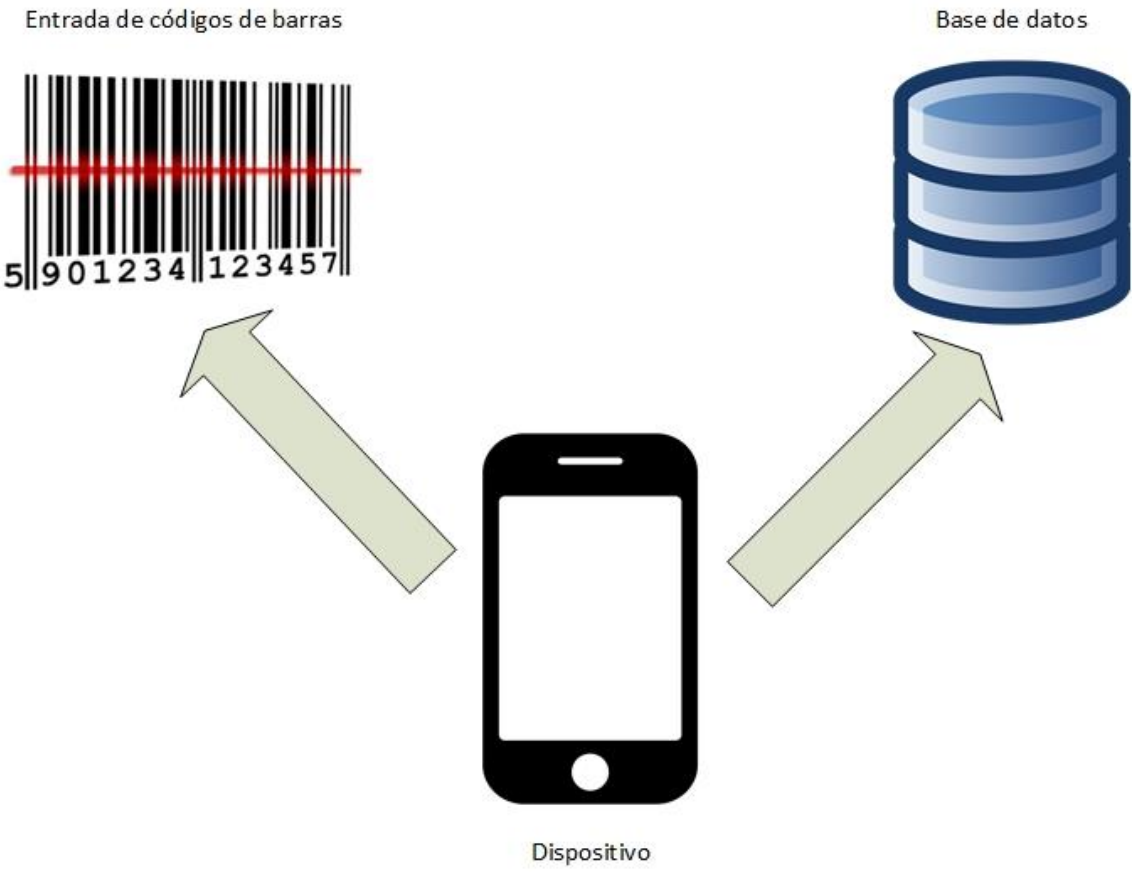


ILUSTRACIÓN 27. COMPONENTES DE LA APLICACIÓN

A continuación, el diagrama de componentes de la aplicación, en la Ilustración 28. En él se representan los principales componentes del sistema así como sus relaciones entre sí. En este caso, la separación de capas de la arquitectura MVC viene representada de la siguiente manera:

- El modelo está determinado por el componente Base de Datos.
- La vista viene determinada por el paquete Interfaz de Usuario.
- El controlador viene determinado por el componente Gestor BD y el paquete Entrada de códigos de barras.

El motivo de utilizar un gestor intermedio es que en futuras versiones la conexión en lugar de realizarse a una base de datos podría ser a un servidor en la nube o mediante

cualquier otro sistema. De este modo se independiza el controlador según el sistema en el que se tenga almacenada la información.

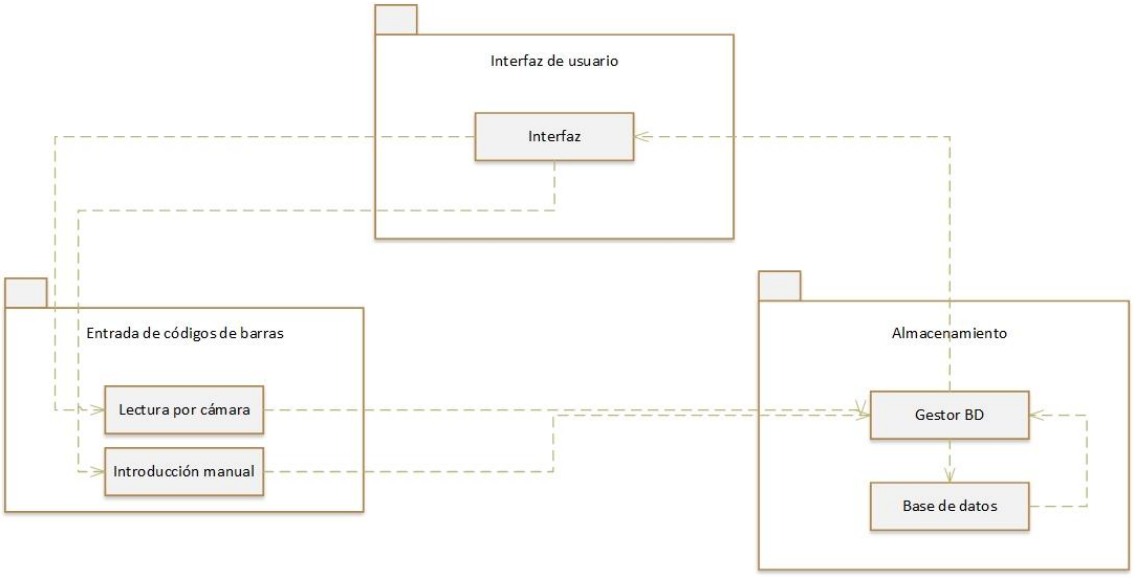


ILUSTRACIÓN 28. DIAGRAMA DE COMPONENTES

Para cada uno de los elementos se realizará su especificación utilizando un formato de tabla que contiene la siguiente información:

- Identificador: identifica de manera única al componente. La nomenclatura será COM_**, donde ** corresponde al número del componente.
- Descripción: descripción del componente.
- Acciones: listado de las acciones que lleva a cabo el componente.
- Dependencias: listado de los componentes de los que depende.
- Requisitos: listado de los requisitos que hacen necesario este componente.

Identificador	COM_01. Interfaz
Descripción	Muestra la información en pantalla y ofrece la interacción con el usuario.
Acciones	- Permite centrar la cámara para hacer captura de códigos. - Muestra información en pantalla. - En cada pantalla, permite acceso a las diferentes opciones.
Dependencias	
Requisitos	RF_01, RF_02, RF_03, RF_04, RF_05, RF_06

Identificador	COM_02. Lectura por cámara
Descripción	Este componente se encarga de realizar lecturas de códigos de barras utilizando la cámara del dispositivo. Esencialmente se utiliza la librería Zebra Crossing (Zxing) de código abierto. Esto evitar la instalación de un software específico para la lectura, aparte de la propia aplicación, lo cual permite una instalación más sencilla.
Acciones	- Utiliza la cámara para hacer captura de códigos. - Recoge el código leído para su utilización. - Recoge el tipo de código leído.
Dependencias	
Requisitos	RF_01

Identificador	COM_03. Introducción manual
Descripción	Permite la introducción manual de códigos en aquellos casos en los que no sea posible la lectura mediante la cámara, ya sea por no disponer de autofoco o por deficiencias en la etiqueta con el código de barras.
Acciones	- Introducción manual del código de barras.
Dependencias	
Requisitos	RF_06

Identificador	COM_04. Gestor BD
Descripción	Recoge la información solicitada por el usuario y realiza los comandos adecuados sobre la base de datos.
Acciones	- Según sea la opción seleccionada por el cliente: o Consulta información sobre la base de datos. o Da de alta un nuevo registro en la base de datos. o Modifica un registro existente en la base de datos. o Elimina un registro existente en la base de datos.
Dependencias	
Requisitos	RF_02, RF_03, RF_04, RF_05

Identificador	COM_05. Base de datos
Descripción	Es la infraestructura donde se va a encontrar la información necesaria de la aplicación.
Acciones	- Interactúa con el Gestor para: o Devolver la información solicitada. o Añadir un nuevo registro. o Modificar un registro. o Eliminar un registro. o Proporciona persistencia de los datos escribiendo en un fichero.
Dependencias	COM_04
Requisitos	RF_02, RF_03, RF_04, RF_05

Tras haber definido los componentes, con el objetivo de evitar que se hayan pasado por alto requisitos tomados durante el análisis, en la Tabla 12 se muestra una matriz de trazabilidad entre requisitos y componentes.

Componentes Requisitos	COM_01	COM_02	COM_03	COM_04	COM_05
RF_01	✓	✓			
RF_02	✓			✓	✓
RF_03	✓			✓	✓
RF_04	✓			✓	✓
RF_05	✓			✓	✓
RF_06	✓		✓		

TABLA 12. MATRIZ DE TRAZABILIDAD COMPONENTES-REQUISITOS

4.3.2. Diseño de la base de datos

Debido a los requisitos de la fase de análisis, es necesario la utilización de un sistema de almacenamiento para poder gestionar la información con la que se va a trabajar en la aplicación. Únicamente se va a almacenar texto, pero se deberán guardar datos correspondientes a los detalles de los dispositivos. En una primera aproximación se utilizarán pocos campos en una única tabla, pero hay que tener en cuenta que en futuras versiones se pueden ampliar el número de campos por los que se pueden realizar búsquedas y realizar índices, y se pueden ampliar el número de tablas para albergar información de usuarios o ubicaciones del cliente, por ejemplo, por lo que se ha optado por utilizar una base de datos, que siempre permite mayor escalabilidad que si se utilizase cualquier otro sistema.

Se van a utilizar las prestaciones que ofrecen Android y SQLite, el sistema gestor de bases de datos que incorpora de fábrica Android, permitiendo acceder a los datos mediante consultas de forma eficaz [30]. Esta base de datos debe contener actualmente la siguiente información, representada en la Ilustración 29.

ILUSTRACIÓN 29. TABLA DE DISPOSITIVOS

La utilización de códigos de barra, que por sus características será de identificadores únicos, permite disponer de una clave primaria para la tabla de dispositivos.

4.3.3. Diseño de la interfaz

En este apartado se muestran tanto el diseño del logotipo del producto como las diferentes interfaces desarrolladas para la aplicación a la vez que se describe su funcionamiento.

El diseño del logotipo se ha creado pensando en su funcionalidad. Un logotipo que sirva de icono de aplicación y que aúne originalidad y abstracción del propio funcionamiento de la aplicación. Por ello, se pensó en utilizar códigos de barra y distintos dispositivos, de una forma suficientemente abstracta para ser genérica pero que nada más verlo mostrase para qué sirve la aplicación. Con esta base como inspiración, la imagen se compone de un código de barras que a su vez da forma, de manera integrada, a diferentes dispositivos, mostrados en su interior, de modo que lo que es leído por un lector de códigos, a su vez es mostrado para los humanos. El resultado final se puede observar en la Ilustración 30.

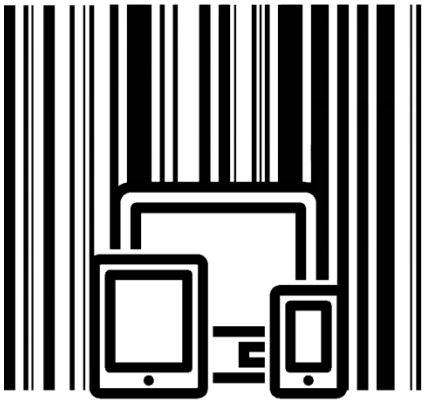
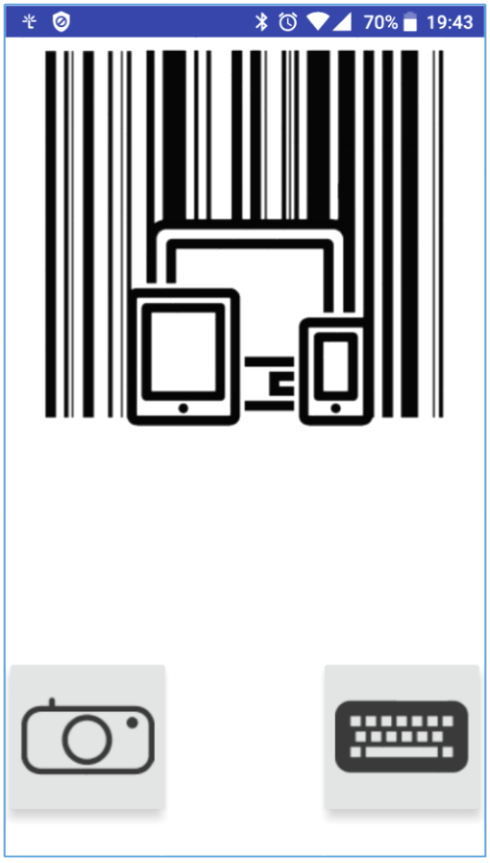


ILUSTRACIÓN 30. LOGOTIPO DE LA APLICACIÓN

El diseño de la aplicación es eminentemente práctico, por lo que no se han incluido gráficos innecesarios y un diseño sobrio, con lo que de paso se cumple el requisito del cliente de obtener el mejor rendimiento posible. Tal y como se menciona en el libro *Android Design Patterns: Interaction Design Solutions for Developers* [31] y en *Android Developers* [32], hay que evitar anidar más de 10 *layouts* (disposiciones gráficas) para

evitar la disminución de rendimiento, de modo que es preferible utilizar *layouts* más planos como *RelativeLayout* o *GridLayout* para mejorar el rendimiento. En casi todas las ocasiones, la disposición utilizada ha sido *RelativeLayout*, por lo que no se ha visto afectado el rendimiento por utilizar varios elementos *LinearLayout*, por ejemplo.

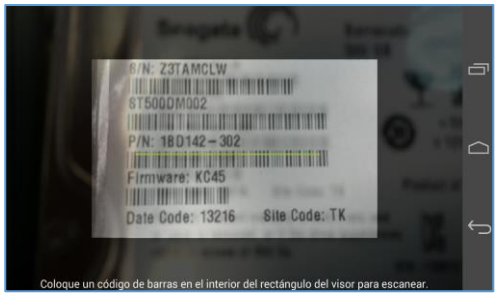


En la pantalla principal se encontrarán las únicas dos opciones que puede realizar el usuario para comenzar: escanear con la cámara o introducir un código manualmente.

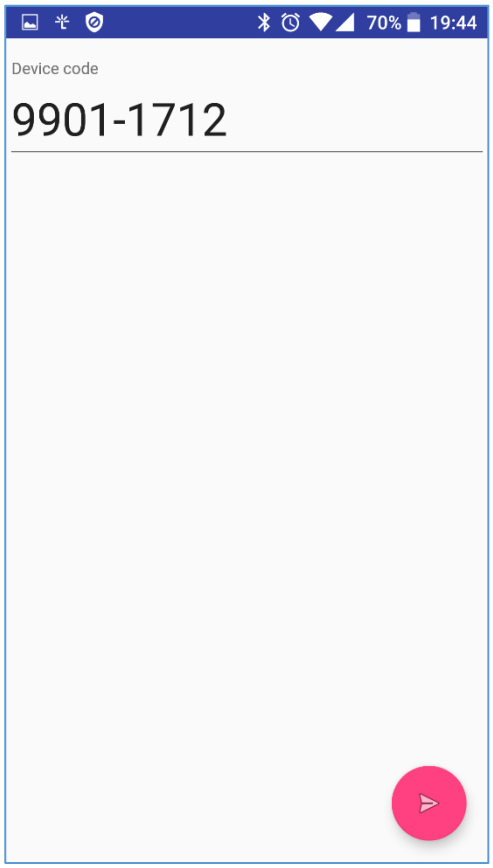
En la parte superior se puede situar una imagen corporativa, como por ejemplo el logo de la empresa.

Además, los botones pueden ser más grandes e incluir algún texto que complemente a la imagen. En este caso se utilizan únicamente imágenes, de modo que un teclado sirve para el botón de introducción manual y una cámara para el botón que realizará el escaneo.

En la interfaz de escáner, se activa la cámara con un marco semitransparente y una guía que ayudan al usuario a encuadrar el código que se desee escanear. La disposición es horizontal para mejorar la captura sobre códigos de barra largos.

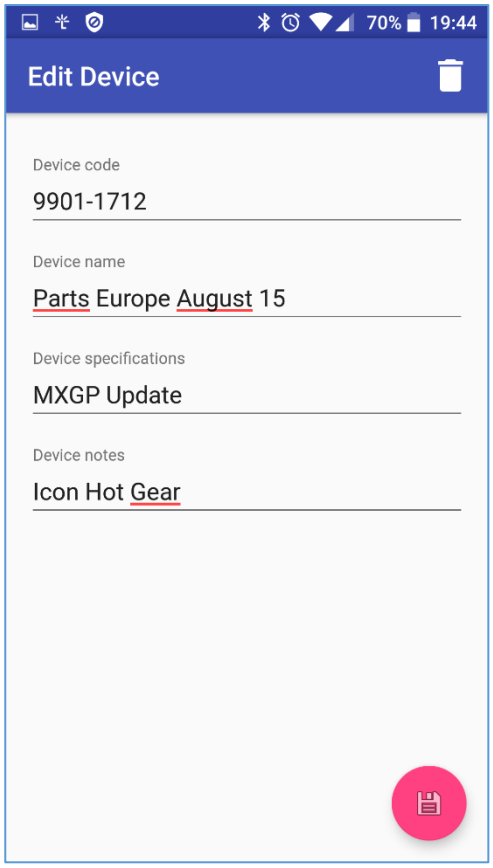


La aplicación auto enfoca y saca una instantánea del código. Como se puede apreciar en la imagen, en caso de existir varios códigos muy cerca unos de otros, es la guía dibujada en la interfaz la que indica cuál de ellos se va a escanear.



Tanto si la lectura se ha realizado con la cámara, como si se ha solicitado realizar la introducción mediante el teclado, la aplicación muestra esta pantalla, donde el campo aparecerá relleno si se usó la cámara o en blanco si se solicitó introducción manual. En ambos casos, se espera que el usuario pulse el botón de enviar, lo cual lanzará la búsqueda en la base de datos. No se puede dejar en blanco el campo del código: la aplicación mostrará un mensaje informando al usuario de que debe rellenar el campo antes de poder continuar.

De este modo, una misma pantalla sirve tanto para la confirmación del código que se ha leído mediante la cámara (podría suceder que estuviese mal leído por falta de luz, código deteriorado…), como para la introducción manual. Con ello aumenta la sencillez y el rendimiento.



Si la búsqueda dio resultado, en la pantalla se muestran los datos obtenidos.

Si no dio resultado, aparece relleno únicamente el código, de modo que los demás campos aparecen en blanco para que el usuario pueda dar de alta un nuevo dispositivo.

Esto sería un prototipo de la vista de una ficha de un código que ya existiría en la base de datos.

El botón guardar, representado por un descriptivo disquete, permite guardar un dispositivo nuevo o actualizar uno existente.

El botón eliminar, representado por una papelera y situado en una posición algo menos utilizada, en la parte superior, permite la eliminación del dispositivo mostrado en pantalla, devolviendo al usuario a la pantalla inicial de la aplicación para continuar con nuevas lecturas.

Por la naturaleza de las aplicaciones Android, es innecesario la creación de botones clásicos como el cierre de una ventana. Los botones de navegación del propio sistema permiten retroceder para volver a escanear o introducir un nuevo código o abandonar la aplicación.

La aplicación contará con ventanas emergentes que mostrarán información sobre el resultado de las operaciones solicitadas, tanto si han sido llevadas a cabo correctamente como si se ha producido algún error.

La secuencia de pantallas se muestra en la Ilustración 31.

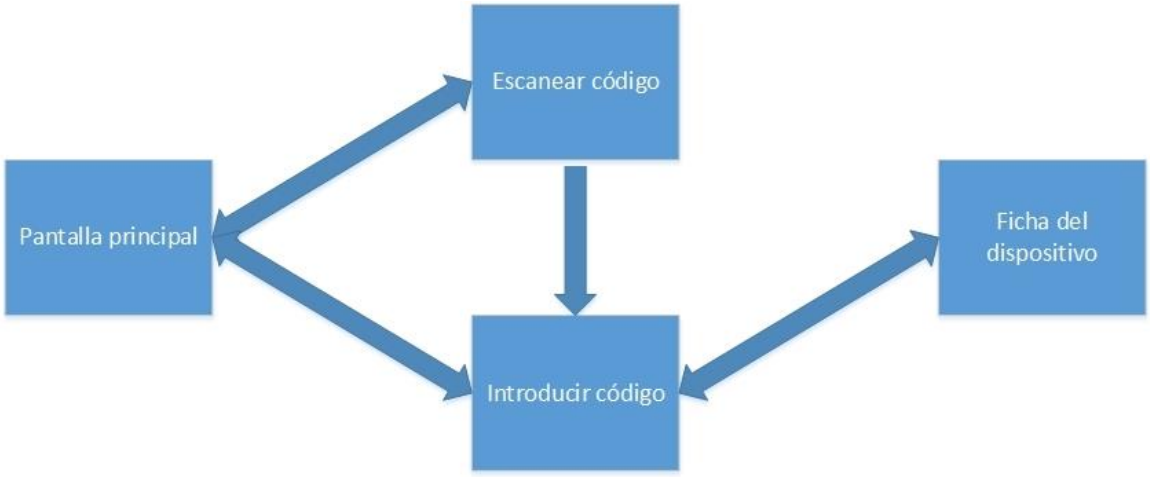


ILUSTRACIÓN 31. NAVEGACIÓN DE PANTALLAS

4.4. Implementación

En este apartado se van a comentar los detalles más relevantes de la implementación de la aplicación, así como las decisiones tomadas durante su desarrollo. No es el objetivo de este documento la creación de un manual de desarrollo extenso, por lo que los aspectos más básicos de implementación relativos al sistema Android quedan fuera del ámbito en el que se encuentra enmarcado. Se pueden encontrar diversos tutoriales y toda la documentación en la web oficial de desarrolladores de Android [12].

Este proceso se dividirá en varias partes que cubrirán diferentes módulos de la implementación:

- Base de datos.
- Lectura mediante cámara.
- Introducción manual.
- Gestión de la base de datos
- Interfaz de usuario.

4.4.1. Implementación de la base de datos

La base de datos es el componente encargado de gestionar el almacenamiento y utilización de los datos de la aplicación. Se ha utilizado el motor de bases de datos incorporado en las bibliotecas Android: SQLite. Es un motor muy popular en la actualidad por ofrecer características muy interesantes como su pequeño tamaño, no necesitar servidor, precisa de poca configuración, es transaccional y por supuesto, es de código libre.

El modo de funcionamiento es simple, ya que no es un proceso independiente, sino que se enlaza con la aplicación pasando a ser parte integral de la misma y de este modo se pueden emplear las funcionalidades SQL a través de llamadas sencillas, reduciendo el tiempo de acceso a la base de datos. Esta base de datos se almacena como un único fichero.

El acceso a los datos se realiza a través de un cursor que se mueve por los registros de las tablas. Cuando se realiza alguna transacción, el fichero se bloquea con un acceso exclusivo.

Según las recomendaciones de la web oficial Android Developers, en el apartado de Almacenar datos en bases de datos SQL [33], para llevar a cabo la implementación de una base de datos SQL en Android, hacen falta tres módulos separados: el *schema* (esquema), el *contract* (la declaración o convenio) y la base de datos.

El esquema es una declaración formal de cómo está organizada la base de datos. El esquema se refleja en declaraciones SQL que se usan para crear la base de datos.

Es muy útil la creación de una clase auxiliar, conocida como clase *contract*, que explícitamente especifica la disposición del esquema de un modo sistemático y autodocumentado.

El código de esta clase *contract* es el siguiente:

```
public class DeviRecDBContract {

    public static abstract class DeviceEntry implements BaseColumns
{

        public static final String TABLE_NAME = "devices";
        public static final String BARCODE = "barcode";
        public static final String NAME = "name";
        public static final String SPECIFICATIONS =
"specifications";
        public static final String NOTES = "notes";
    }
}
```

La clase *contract* es un contenedor para las constantes que definen nombres para URIs (*Uniform Resource Identifier*, Identificador Uniforme de Recursos), tablas y columnas. La clase *contract* permite usar las mismas constantes a través de todas las demás clases en el mismo paquete. Esto permite cambiar el nombre de una columna en un único lugar y propagarlo por todo el código.

Un buen modo de organizar una clase *contract* es poner las definiciones que son globales a toda la base de datos en el nivel raíz de la clase. Después crear una clase interna por cada tabla que enumera sus columnas.

Una vez que se ha definido qué aspecto tiene la base de datos, hay que implementar métodos que creen y mantengan la base de datos y las tablas.

Android almacena la base de datos en el área privada de espacio de disco asociada a la aplicación, lo cual hace que los datos estén seguros, porque por defecto esta área no es accesible a otras aplicaciones.

Un útil set de APIs está disponible en la clase *SQLiteOpenHelper*. Cuando se usa esta clase para obtener referencias a la base de datos, el sistema lleva a cabo operaciones potencialmente de larga duración de crear y actualizar la base de datos sólo cuando se necesita y no durante el arranque de la aplicación. Todo lo que se necesita es llamar a los métodos *getWritableDatabase()* o *getReadableDatabase()*. Es por ello que se debe crear una clase que extienda de SQLiteOpenHelper, cuyo nombre será DeviRecDBHelper. Esta clase permite, entre otras cosas, aislar sentencias SQL de definiciones, como por ejemplo:

Lo que en SQL sería:

```
CREATE TABLE Devices (KEY _ID, UNIQUE BARCODE, TEXT NAME, TEXT
SPECIFICATIONS, TEXT NOTES)
```

Se traduce como una constante *sqlCreate*:

```
private static final String sqlCreate = "CREATE TABLE "+
DeviceEntry.TABLE_NAME + " ("
            + DeviceEntry.BARCODE + " UNIQUE,"
            + DeviceEntry.NAME + " TEXT,"
            + DeviceEntry.SPECIFICATIONS + " TEXT,"
            + DeviceEntry.NOTES + " TEXT"
            + ")";
```

De este modo, se definen también las sentencias que permiten crear, eliminar, modificar y consultar registros.

```
public Cursor getDeviceByBarcode(String deviceBarcode){
Cursor c = getReadableDatabase().query(
                DeviceEntry.TABLE_NAME,
                null, //columns
                DeviceEntry.BARCODE + " LIKE ?",
                new String[]{deviceBarcode},
                null,
                null,
                null);
```

```
        return c;
}

public int deleteDevice(String deviceBarcode) {
    return
                getWritableDatabase().delete(DeviceEntry.TABLE_NAME,
                        DeviceEntry.BARCODE + " LIKE ?",
                        new String[]{deviceBarcode});
}

public int updateDevice(DeviRecDevice device, String deviceBarcode)
{
        return getWritableDatabase().update(
                DeviceEntry.TABLE_NAME,
                device.toContentValues(),
                DeviceEntry.BARCODE + " LIKE ?",
                new String[]{deviceBarcode}
        );
}

public long saveDevice(DeviRecDevice device) {
        SQLiteDatabase db = getWritableDatabase();
        return db.insert(DeviceEntry.TABLE_NAME, null,
device.toContentValues());
}
```

Para manejar correctamente los datos, es necesaria la creación de una clase que tome los datos de la base de datos y los convierta en un objeto que permita su manipulación. Es por ello que se crea la clase DeviRecDevice. En ella destaca la sobrecarga de la clase constructora para sus diferentes usos (objeto con todos sus atributos, objeto nuevo a partir de únicamente un código de barras y objeto a partir de un cursor que devuelve una dirección a la base de datos al realizar una consulta) así como la traducción de atributos en campos de la base de datos mediante pares de valores.

```
public class  DeviRecDevice {
    private String barcode = "";
    private String name = "";
```

```
    private String specifications = "";
    private String notes = "";

    public DeviRecDevice (String barcode, String name,
                          String specifications, String notes ){
        this.barcode = barcode;
        this.name = name;
        this.specifications = specifications;
        this.notes = notes;
    }

    public DeviRecDevice (String barcode) {
        this.barcode = barcode;
        this.name = "";
        this.specifications = "";
        this.notes = "";
    }

    public DeviRecDevice (Cursor cursor) {
        barcode =
cursor.getString(cursor.getColumnIndex(DeviceEntry.BARCODE));
        name =
cursor.getString(cursor.getColumnIndex(DeviceEntry.NAME));
        specifications =
cursor.getString(cursor.getColumnIndex(DeviceEntry.SPECIFICATIONS));
        notes =
cursor.getString(cursor.getColumnIndex(DeviceEntry.NOTES));
    }
    //translate pairs into values
    public ContentValues toContentValues() {
        ContentValues values = new ContentValues();
        values.put(DeviceEntry.BARCODE, barcode);
        values.put(DeviceEntry.NAME, name);
        values.put(DeviceEntry.SPECIFICATIONS, specifications);
        values.put(DeviceEntry.NOTES, notes);
        return values;
    }}
```

4.4.2. Implementación del módulo de lectura

Para la implementación del módulo de lectura utilizando la cámara del dispositivo se ha utilizado la librería de código libre Zxing (abreviatura de *Zebra Crossing*) [34].

Esta librería soporta los siguientes formatos en 1D y 2D (una dimensión y dos dimensiones):

1D product	1D industrial	2D
UPC-A	Code 39	QR Code
UPC-E	Code 93	Data Matrix
EAN-8	Code 128	Aztec (beta)
EAN-13	Codabar	PDF 417 (beta)
	ITF	
	RSS-14	
	RSS-Expanded	

TABLA 13. FORMATOS SOPORTADOS POR ZXING

La integración de librerías no carece de dificultad, principalmente cuando se utilizaba Eclipse como IDE (Integrated Development Environment, entorno de desarrollo integrado), aunque en sucesivas versiones de Android Studio se está mejorando considerablemente en este sentido. Actualmente, aunque se realiza mediante un sistema algo oscuro, es bastante sencillo. Hay que especificar las dependencias y el repositorio Maven [35] en el archivo *build.gradle* dentro del directorio *app* del proyecto.

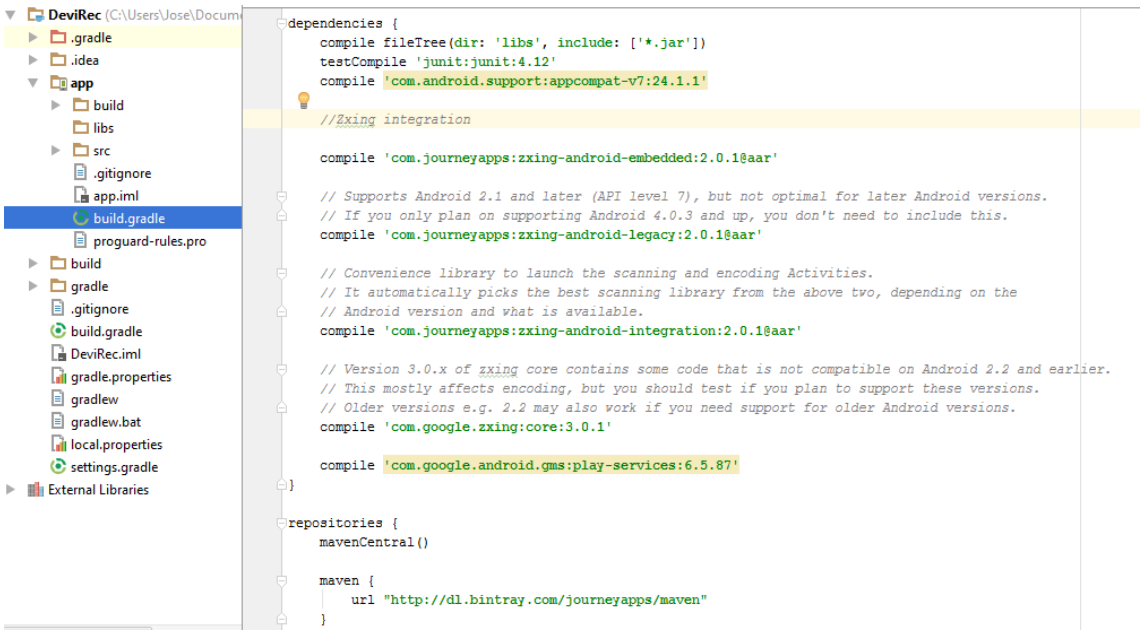


ILUSTRACIÓN 32. INTEGRACIÓN DE ZXING EN EL PROYECTO

Como se puede apreciar en la Ilustración 32, se añaden varias líneas a los apartados *dependencies* y las referencias a *Maven* en *repositories* del archivo *build.gradle* contenido en el directorio *app*, que es el específico de la aplicación. El uso de una dependencia u otra está determinado por la versión de Android en la que se pretenda ejecutar la librería [36].

La utilización de la librería una vez integrada es simple: se utilizan IntentIntegrator e IntentResult.

El primero inicia el escaneo mediante la cámara.

```
//instantiate ZXing integration class
IntentIntegrator scanIntegrator = new IntentIntegrator(this);
//start scanning
scanIntegrator.initiateScan();
```

El segundo recoge el resultado del escaneo para permitir trabajar con él. Este resultado contiene el código escaneado y su formato.

```
//retrieve result of scanning - instantiate ZXing object
IntentResult scanningResult =
IntentIntegrator.parseActivityResult(requestCode, resultCode,
intent);
//check we have a valid result
if (scanningResult != null) {
    //get content from Intent Result
    String scanContent = scanningResult.getContents();
    //get format name of data scanned
    String scanFormat = scanningResult.getFormatName();
    //output to UI
    formatTxt.setText(getString(R.string.text_format) + scanFormat);
    contentTxt.setText(getString(R.string.text_content) +
scanContent);
```

4.4.3. Implementación del módulo de introducción manual de códigos

Este módulo ha sido realizado pensando en su reutilización dentro de la propia aplicación, utilizándose para dos cometidos: por un lado es el destino del código leído por el módulo lector de códigos de barras, a modo de confirmación de lo leído, ya que debido a condiciones de disposición del código de barras o de luminosidad, el código leído podría no ser correcto; por otro es el módulo al que se llega si en la pantalla principal se selecciona la introducción manual del código de barras.

En caso de venir desde la lectura por la cámara, el código aparece ya introducido y sólo hay que confirmar que es correcto para pasar a su gestión. En caso de venir desde la opción de lectura manual, el campo del código aparece vacío listo para rellenarlo y pasar a su gestión.

```
etCode = (TextInputEditText) this.findViewById(R.id.et_code);

etCode.setText(null);

Intent i = getIntent();// Get the intent
```

```
if (i != null) {

   mDeviceBarcode = i.getStringExtra("CONTENT");
   etCode.setText(mDeviceBarcode);
   }
   else {
         mDeviceBarcode = "";
         etCode.setText(mDeviceBarcode);

etCode.setHint(getApplicationContext().getString(R.string.et_code_hi
nt));
}
```

Ya sea porque se ha tomado el código desde la cámara o desde la introducción manual, el botón de envío permite su paso al siguiente módulo, donde se realizará la gestión del código. Así mismo, vigila para que el código, al menos, tenga un valor distinto de vacío en el momento de su tramitación, por lo que si la longitud es cero, muestra un mensaje emergente avisando de ello y no se envía ninguna información.

```
FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab_send_barcode);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (etCode.getText().toString().length() != 0) {
                    // Send Barcode to next activity
                    Intent i = new Intent(getApplicationContext(),
AddEditDeviceActivity.class);
                    i.putExtra
("CONTENT",etCode.getText().toString());
                    startActivity(i);
                }
                else {
Toast.makeText(getApplicationContext(),getApplicationContext().getSt
ring(R.string.required_field),Toast.LENGTH_LONG).show();
                }
    }});
```

4.4.4. Implementación de la gestión de fichas de dispositivos

Este módulo gestiona la información recibida y, según sus características, trabaja con la base de datos y la interfaz para ofrecer los resultados esperados.

Primero se recoge el código de barras. La forma más adecuada es utilizando un paquete, en lugar de enviar cada información por separado. Esto lo realiza un objeto de la clase Bundle, que permite recoger pares de valores para su posterior manejo.

```
public static AddEditDeviceFragment newInstance(String
mDeviceBarcode) {
        AddEditDeviceFragment fragment = new
AddEditDeviceFragment();
        Bundle args = new Bundle();
        args.putString("CONTENT", mDeviceBarcode);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mDeviceBarcode = getArguments().getString("CONTENT");
        }

    }
```

Después se realiza una consulta a la base de datos para comprobar si es un código que ya existe o se trata de un nuevo código. Los accesos a la base de datos siempre se van a realizar mediante tareas asíncronas, hilos diferentes de ejecución que permiten mayor agilidad general. Ello se realiza extendiendo las clases de *AsyncTask*.

```
private class GetDeviceByIdTask extends AsyncTask<Void, Void,
Cursor> {
  @Override
  protected Cursor doInBackground(Void... voids) {
    return mDeviRecDBHelper.getDeviceByBarcode(mDeviceBarcode);
  }
  @Override
  protected void onPostExecute(Cursor cursor) {
   if (cursor != null && cursor.moveToLast()) {
      isNewDevice = false;
      showDevice(new DeviRecDevice(cursor));
   }else {
      isNewDevice = true;
      showDevice(new DeviRecDevice(mDeviceBarcode));
   }
  }
}
```

Una vez que se sabe si el dispositivo es uno nuevo o uno ya existente, se muestran o bien todos los datos o bien únicamente relleno el campo del código de barras para permitir que el usuario agregue la información que estime oportuna.

En el momento de guardar, realmente se comprueba si el dispositivo ya existía en cuyo caso la opción de guardar realiza una tarea de actualización del registro en la base de datos y se añade registro nuevo si el código de barras no se encontraba previamente en la base de datos.

```
private class AddEditDeviceTask extends AsyncTask<DeviRecDevice,
Void, Boolean> {

  @Override
  protected Boolean doInBackground(DeviRecDevice... deviRecDevices)
{
     if (isNewDevice == false) {
     return mDeviRecDBHelper.updateDevice(deviRecDevices[0],
mDeviceBarcode) > 0;
     } else {
```

```
            return mDeviRecDBHelper.saveDevice(deviRecDevices[0]) > 0;
              }
}
```

El borrado funciona de modo similar, eliminando la ficha del dispositivo que actualmente se encuentra en pantalla y posteriormente regresando a la pantalla principal, de modo que se elimina la pila y se evita que al pulsar el botón de vuelta a atrás en Android, se visualicen datos en pantalla ya eliminados y regresando a un estado más lógico, como es el comienzo de la aplicación, por si el usuario pretende salir (pulsa atrás una sola vez y sale, en lugar de tener que pulsar varias veces pasando por pantallas previas) o si pretende volver a escanear cualquier otro código.

```
private class DeleteDeviceTask extends AsyncTask<Void, Void,
Integer> {

        @Override
        protected Integer doInBackground(Void... voids) {
            return mDeviRecDBHelper.deleteDevice(mDeviceBarcode);
        }

        @Override
        protected void onPostExecute (Integer integer) {

            Intent backToStart = new Intent
(getActivity(),MainActivity.class);
            backToStart.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(backToStart);

        }

    }
```

4.4.5. Implementación de la interfaz de usuario

La última versión de Android Studio contiene una herramienta visual que permite ver el resultado en pantalla, ya disponible en versiones anteriores, con algo disponible desde su versión 2.2, que al mismo tiempo muestra su correspondencia con los elementos definidos mediante XML.

El XML, siglas del inglés *eXtensible Markup Language*, es un Lenguaje de Etiquetado Extensible definido por el World Wide Web Consortium [37]. El XML sirve para estructurar, almacenar e intercambiar información.

En la Ilustración 33 se puede observar el esquema de la pantalla principal de la aplicación.

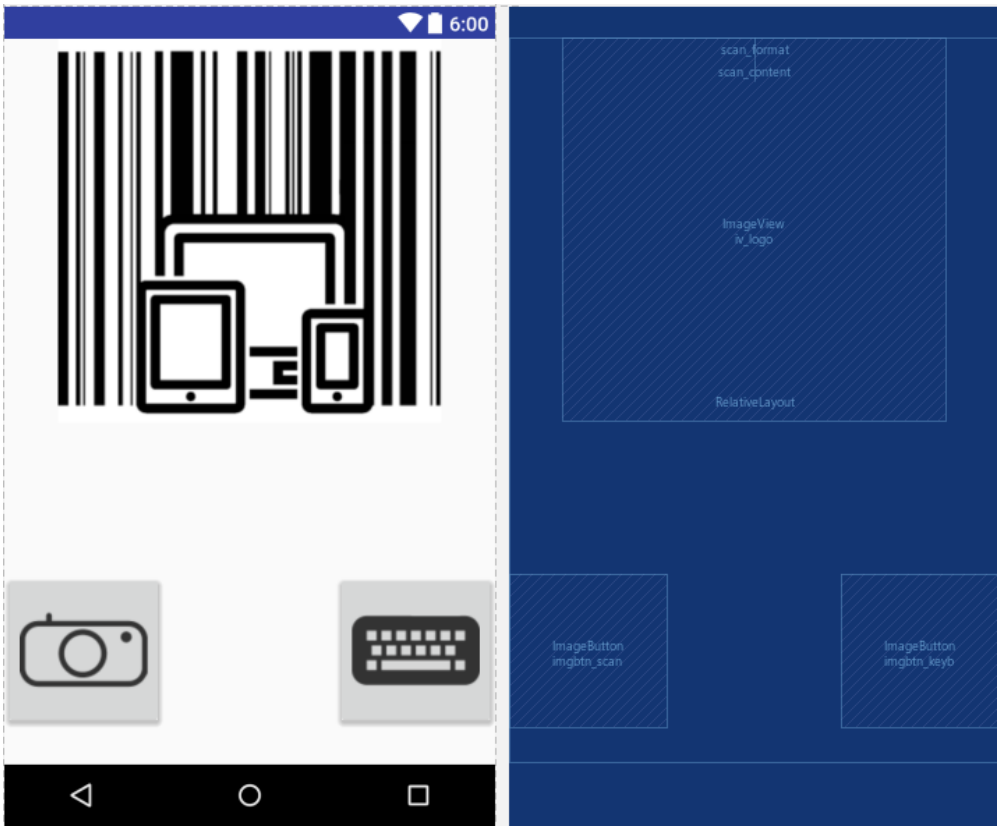


ILUSTRACIÓN 33. ESQUEMA VISUAL DE LA PANTALLA PRINCIPAL

Internamente se compone de una disposición relativa (RelativeLayout), en la que todos los elementos se disponen en la pantalla en relación al contenedor y al resto de los componentes.

El código XML que permite mostrar la disposición de la Ilustración 33, es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".MainActivity" >

    <TextView
        android:id="@+id/scan_format"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textIsSelectable="true"
        android:layout_centerHorizontal="true"
        />

    <TextView
        android:id="@+id/scan_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textIsSelectable="true"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/scan_format" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
app:srcCompat="@drawable/icondevirecfondoblanco300"
        android:id="@+id/iv_logo"
        android:src="@drawable/icondevirecfondoblanco300"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
app:srcCompat="@drawable/cameraicon100"
        android:src="@drawable/cameraicon100"
        android:id="@+id/imgbtn_scan"
        android:elevation="8dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="29dp" />

    <ImageButton
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
app:srcCompat="@drawable/keyboardicon100"
        android:src="@drawable/keyboardicon100"
        android:id="@+id/imgbtn_keyb"
        android:elevation="8dp"
        android:layout_alignTop="@+id/imgbtn_scan"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Esencialmente lo componen una imagen y dos botones con imagen, que además tienen efecto de elevación, para resaltar su función como botón.

Todos los elementos necesitan de una identificación única, definida mediante el uso de la etiqueta *android:id*. Esta identificación permite referenciarlos en el código posteriormente, de modo que se pueden realizar modificaciones sobre ellos en tiempo de ejecución.

La pantalla de confirmación del código leído, que a su vez es la de introducción manual de código se compone de dos archivos XML, que ofrecen dos capas: la que engloba elementos como el menú superior y el botón flotante y el contenido.

El contenido dispone de un tipo de campo de texto con cierta animación que lo hace atractivo visualmente, el TextInputLayout, que a diferencia de los clásicos campos estáticos EditText dispone de una animación que permite que el texto por defecto de un campo, que sirve para dar un indicio o insinuar (de hecho la propiedad es *hint*, que significa en inglés indicio o insinuación), se desplace y se convierta en el título del campo una vez el usuario pulsa sobre el campo para escribir.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:descendantFocusability="beforeDescendants"
    android:focusableInTouchMode="true"
    android:orientation="vertical"
    tools:context=".BarcodeActivity"
    tools:showIn="@layout/activity_barcode">
```

```
    <android.support.design.widget.TextInputLayout
        android:id="@+id/til_barcode"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/activity_vertical_margin">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/et_code"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/et_code_hint"

android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:imeOptions="actionDone"
            android:maxLines="1"
            android:maxLength="16"
        />
    </android.support.design.widget.TextInputLayout>
</LinearLayout>
```

El contenedor dispone de un botón de acción flotante (*Floating Action Button* en inglés, más utilizado en su forma abreviada “fab”). Que en las últimas versiones de Android se encuentra en el lugar más destacado, la parte inferior derecha de la pantalla, representando la acción más utilizada y que el usuario realizará con mayor probabilidad. El contenedor debe tener una referencia al contenido, mediante la etiqueta *<include>*, cuyo nombre de archivo es *content_barcode*.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".BarcodeActivity" android:id="@+id/fab_send">

    <include layout="@layout/content_barcode"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_send_barcode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_menu_send"/>
</android.support.design.widget.CoordinatorLayout>
```

Por último, la pantalla que muestra, edita y/o elimina está compuesta por un contenedor y un contenido de tipo *fragment*.

El contenedor, como en el caso anterior, contiene un marco que incluye el menú superior, donde estará el botón de borrado, y el botón de acción flotante, que será el de guardado o actualización, y una referencia al contenido.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"

tools:context="com.darthyandros.devirec.addeditdevice.AddEditDeviceA
ctivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar_edit"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_add_edit_device" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab_save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_menu_save" />

</android.support.design.widget.CoordinatorLayout>
```

El contenido será el responsable de mostrar todos los detalles del dispositivo. Por ello, en primer lugar, envolviendo al resto de etiquetas, se encuentra el ScrollView, que permite que su contenido pueda desplazarse arriba y abajo en caso necesario. La versión actual sólo dispone de cuatro campos, pero posteriores versiones pueden tener nuevos campos que requerirán desplazar la pantalla para poder ser consultados o modificados.

Todos los campos contenidos se han realizado mediante TextInputLayout, lo cual permite darle a los campos del formulario cierto dinamismo, superior al caso de haber utilizado EditText, completamente estáticos.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollViewAddEdit"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:descendantFocusability="beforeDescendants"
    android:focusableInTouchMode="true"
    android:orientation="vertical"

tools:context="com.darthyandros.devirec.addeditdevice.AddEditDeviceF
ragment">

    <android.support.design.widget.TextInputLayout
        android:id="@+id/til_barcode"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginTop="@dimen/activity_vertical_margin">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/et_code"
            android:hint="@string/et_code_hint"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="text"/>
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/til_name"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:layout_marginTop="@dimen/activity_vertical_margin">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/et_name"
            android:hint="@string/et_name_hint"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="text"/>
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/til_specifications"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/activity_vertical_margin">
```

```
        <android.support.design.widget.TextInputEditText
            android:id="@+id/et_specifications"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:hint="@string/et_specifications_hint"
            android:inputType="text"/>
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/til_notes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/activity_vertical_margin">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/et_notes"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:hint="@string/et_notes_hint"
            android:inputType="text"/>
    </android.support.design.widget.TextInputLayout>
   </LinearLayout>
</ScrollView>
```

Como detalle adicional, si se rotaba la pantalla mientras se estaban introduciendo o modificando datos en los diferentes campos, la información se reiniciaba. Esto es debido al funcionamiento de las aplicaciones en Android, que por defecto, al rotar, vuelven a crearse, manteniendo solamente aquellos datos que fuesen persistentes, como por ejemplo aquellos previamente guardados, o los títulos de los campos.

Para evitar esto, en el manifiesto, el XML principal de declaración que incluye el resto de actividades que forman parte de la aplicación así como la declaración de permisos que necesita la aplicación y otros datos y parámetros de la aplicación en general, hay que añadir un parámetro a la actividad que no se desea reiniciar al rotar para evitar perder cualquier información introducida. Esto se hará en la actividad AddEditDeviceActivity, mediante *Android:configChanges="screenSize|orientation"*.

A continuación se muestra el archivo *Manifest* completo. Éste contiene información sobre la versión de la aplicación, el nombre de la aplicación, los permisos que se solicitarán al usuario, necesarios para el funcionamiento de la aplicación (en este caso, acceso a la cámara y al almacenamiento interno) y las diferentes actividades que se podrán utilizar con la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.darthyandros.devirec">
    android:versionCode="1"
    android:versionName="1.0">
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <uses-feature android:name="android.hardware.camera2"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/devirec_icon"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"
            android:launchMode="singleTask">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category
android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity
            android:name=".devices.DevicesActivity"
            android:label="@string/title_activity_devices"
            android:theme="@style/AppTheme.NoActionBar"/>
        <activity
            android:name=".devicedetail.DeviceDetailActivity"
            android:label="@string/title_activity_device_detail"
            android:theme="@style/AppTheme.NoActionBar"/>

        <activity
            android:name=".addeditdevice.AddEditDeviceActivity"
            android:label="@string/title_activity_add_edit_device"
            android:theme="@style/AppTheme.NoActionBar"
            android:configChanges="screenSize|orientation"/>

        <activity
            android:name=".BarcodeActivity"
            android:label="@string/title_activity_barcode"
            android:theme="@style/AppTheme.NoActionBar">
        </activity>
    </application>

</manifest>
```

Capítulo 5. Conclusiones y líneas futuras

5.1. Conclusiones

El desarrollo de este proyecto no ha carecido de dificultad. Se ha tenido que superar una serie de retos y valorar diferentes decisiones, estudiando cuáles serían las más correctas para completar correctamente su desarrollo. Lo que se tenía claro desde el principio era la respuesta a la pregunta "¿Qué se podría hacer a nivel informático para mejorar el rendimiento de una empresa real dada?". La creación de una herramienta que facilitase y mejorase el trabajo diario, desde la experiencia de la observación de su forma de trabajar.

El estudio previo al proyecto sitúa a éste en un contexto en el que permite, con un alto grado de fiabilidad, decidir sobre su viabilidad, incluso antes de comenzar con la gestión del proyecto. Es fundamental la revisión del mercado actual desde diferentes puntos de vista. Es importante comparar el proyecto con otros existentes y verificar si lo que se quiere hacer dispone de un nicho de mercado explotable o no. Al mismo tiempo, es importante la decisión de en qué plataforma se debería implementar, puesto que en gran medida determina el éxito o el fracaso, así como los ingresos que puede generar la aplicación. En definitiva, un estudio de estas características supone una inversión inicial económicamente interesante, que evalúa los riesgos y permite comprobar si realmente merece la pena invertir esfuerzos y dinero en la realización de un proyecto determinado, con lo que se convierte en algo indispensable como paso inicial en cualquier proyecto serio.

Dados los resultados ofrecidos por este estudio inicial, a continuación se pudo dar paso a la gestión de proyecto, análisis, diseño e implementación de la aplicación propuesta,

puesto que ocuparía un lugar en el mercado todavía no ocupado por ningún otro producto y puede suponer una ventaja ante la competencia y por lo tanto obtener ingresos con ello.

Una vez que se decidió utilizar la plataforma Android como una de las consecuencias del estudio inicial, se originaron dos retos: conocer la arquitectura de la plataforma Android y trabajar con el lenguaje de programación Java, ambos desconocidos totalmente por el desarrollador de este proyecto. Los conocimientos generales en programación orientada a objetos, bases de datos y sistemas operativos, adquiridos durante la carrera, sumados a un gran esfuerzo empleado en recabar documentación y referencias de programación en Android y Java, más las numerosas horas de programación que finalmente dieron forma a la aplicación, fueron fundamentales para poder realizar correctamente este apartado.

Comenzar desde prácticamente cero conocimientos en una determinada plataforma, compaginando estudios y trabajo, salir adelante y conseguir un objetivo. No ha sido fácil. Pero la recompensa es doble. Como se suele decir, sin dificultad no habría gloria en la victoria. Es eso exactamente lo que se ha conseguido: una gran satisfacción personal al poderlo conseguir.

5.2. Líneas futuras

Tras haber finalizado este proyecto, existen diferentes mejoras que podrían marcar las líneas a seguir en el futuro de la aplicación desarrollada. A continuación se detallan una serie de posibles líneas de trabajo que han ido surgiendo durante el propio desarrollo llevado a cabo, ya sea por sugerencias o comentarios del cliente, ya sea como ocurrencias del propio desarrollador una vez observados los resultados y otras tecnologías y aplicaciones, que incluso siendo de muy diferentes ámbitos, siempre pueden aportar ideas de por dónde pueden encaminarse futuras versiones. La aplicación básica desarrollada se diseñó con vistas a llevar a cabo estas y otras ampliaciones, de modo que sería fácilmente escalable y mejorable debido a los módulos utilizados.

Respecto al lector de códigos, se proponen las siguientes posibilidades:

- Integrar completamente la lectura en la aplicación, de modo que se pudiese tener media pantalla con la interfaz de lectura y la otra media pantalla con el resultado leído o la ficha de producto leído. Esto sería posible integrando la librería Z-Xing mediante *fragments*.
- Adjuntar ubicaciones GPS a las capturas de códigos realizadas, al estilo en que lo hacen aplicaciones como Google Maps con las fotos, de modo que se pudiese asignar un código de producto a un cliente de la zona fácilmente, en lugar de introducirlo manualmente.

Respecto a la base de datos, se podrían avanzar las siguientes características:

- Ampliar la base de datos actual para guardar información de clientes, de modo que se relacione qué clientes tienen qué productos. Sumado a la característica del GPS, se podría hacer casi automáticamente.

- Ampliar el número de campos utilizados para definir un producto. Actualmente son pocos los campos, a modo de demostración de lo que se puede realizar, pero lo ideal es que existiesen campos más descriptivos, propios del tipo de productos a almacenar, tales como: CPU, velocidad, RAM, capacidad, etc... Indexando los principales de modo que se puedan hacer búsquedas.
- Ampliar la base de datos con información propia de intervenciones o incidencias, guardando también información del técnico que realizó la intervención y pudiendo realizar búsquedas por técnico o por intervención.
- Utilizar una base de datos en la nube, para lo cual harían falta nuevos permisos en la aplicación de acceso a internet, pero se estaría utilizando una base de datos mucho más dinámica y multiusuario.

6. Bibliografía y referencias

[1] «GLPi Project,» [En línea]. Available: http://www.glpi-project.org/.

[2] «Retail Inventory,» 2016. [En línea]. Available: https://ww2.cashierlive.com/inventory.

[3] «Almyta,» 2016. [En línea]. Available: http://www.almyta.com/abc_inventory_software.asp.

[4] «Partkeepr,» 2016. [En línea]. Available: https://partkeepr.org/.

[5] Kantar Worldpanel, «Smartphone OS market share,» Diciembre 2015. [En línea]. Available: http://www.kantarworldpanel.com/global/smartphone-os-market-share/.

[6] El Confidencial, «BlackBerry se pasa a Android,» 25 septiembre 2015. [En línea]. Available: http://www.elconfidencial.com/tecnologia/2015-09-25/priv-android-blackberry-smartphone_1037071/.

[7] Apps User, «Firefox OS,» 9 diciembre 2015. [En línea]. Available: http://appsuser.net/www/2015/12/09/firefox-os/.

[8] Xataka, «Ecosistema Microsoft en Android,» 4 febrero 2016. [En línea]. Available: http://www.xataka.com/servicios/microsoft-esta-armando-un-ecosistema-brutal-en-android-e-ios.

[9] «Open Handset Alliance,» 2007. [En línea]. Available: http://www.openhandsetalliance.com/oha_overview.html.

[10] Xataka, «Fecha de caducidad de los smartphones,» 14 noviembre 2014. [En línea]. Available: http://www.xataka.com/moviles/asi-es-la-vida-tu-smartphone-tiene-fecha-de-caducidad.

[11] Android Police, «Google To Begin Forcing OEMs To Certify Android Devices With A Recent OS Version If They Want Google Apps,» 10 febrero 2014. [En línea].

Available: http://www.androidpolice.com/2014/02/10/rumor-google-to-begin-forcing-oems-to-certify-android-devices-with-a-recent-os-version-if-they-want-google-apps/.

[12] Android Developers, «Dashboards,» 2016. [En línea]. Available: http://developer.android.com/about/dashboards/index.html.

[13] Eclipse, [En línea]. Available: https://eclipse.org/.

[14] O. Cinar, Android Apps with Eclipse, Apress, 2012.

[15] N. Smyth, Android Studio Development Essentials, 2015.

[16] Android Developers, «Android Studio 2 Preview,» 23 noviembre 2015. [En línea]. Available: http://android-developers.blogspot.com.es/2015/11/android-studio-20-preview.html.

[17] G. Fernández Pérez, iOS, Todo lo que siempre has querido saber sobre tu iPhone y iPad, 2013.

[18] Apple Developers, «Xcode 7,» 2016. [En línea]. Available: https://developer.apple.com/xcode/.

[19] E. Flores Gonzalo, Aprende a Programar Swift: Programación iOS. 2ª Edición, 2016.

[20] M. Á. G. Arias, Programación en Objective-C: Programa para Iphone y MAC, 2013.

[21] N. Smyth, iOS 9 App Development Essentials: Learn to develop iOS 9 apps using Xcode 7 and Swift 2, 2015.

[22] Developer Economics, [En línea]. Available: https://www.developereconomics.com.

[23] E. S. Norman, S. A. Brotherton y R. T. Fried, Work Breakdown Structures: The Foundation for Project Management Excellence, John Wiley & Sons, 2010.

[24] D. Haughley, «Project Management Tools. Project Smart.,» 2015. [En línea]. Available: https://www.projectsmart.co.uk/project-management-tools.php.

[25] Android Developers, «Android Studio,» [En línea]. Available: https://developer.android.com/studio/index.html.

[26] «Java,» [En línea]. Available: www.java.com.

[27] «Eclipse,» [En línea]. Available: https://eclipse.org/.

[28] «Paint.Net,» [En línea]. Available: http://www.getpaint.net/.

[29] IEEE, «IEEE.org,» [En línea]. Available: https://standards.ieee.org/findstds/standard/830-1998.html.

[30] B. Phillips, C. Stewart, B. Hardy y K. Marsicano, Programación Con Android, 2016.

[31] G. Nudelman, Android Design Patterns: Interaction Design Solutions for Developers, Wiley, 2013.

[32] Android Developers, «Optimizing Layout Hierarchies,» 2016. [En línea]. Available: https://developer.android.com/training/improving-layouts/optimizing-layout.html.

[33] Android Developers, «Saving Data in SQL Databases,» [En línea]. Available: https://developer.android.com/training/basics/data-storage/databases.html.

[34] «Zxing Project,» [En línea]. Available: https://github.com/zxing/zxing.

[35] M. O'Brien, Maven: The Complete Reference, 2016.

[36] K. Kousen, Gradle Recipes for Android, O'Reilly, 2016.

[37] T. Boulanger, XML Práctico. Bases Esenciales, Conceptos Y Casos Prácticos, Eni, 2015.